実用インターフェース設計法

マイコン活用のためのハードウェア技術入門

畔津明仁 著

●良い製品には良い設計思想が生かされており，良い設計思想には，必ずしっかりとしたバックグラウンドがあります．

●ますますIC/LSI化が進み高度化するエレクトロニクス技術も，バックグラウンドとなる技術さえあれば怖くありません．

●CORE BOOKSは，あなたのエレクトロニクス技術のバックグラウンドづくりを応援するCQ出版社の新しい書籍シリーズです．

《カバー》

●デザイン／宰　良二

●フォト　／佐瀬　真

実用インターフェース設計法

マイコン活用のためのハードウェア技術入門

畔津明仁 著

CQ出版社

まえがき

わたしたちが少し複雑な機能を持つ装置を設計するとき，その中にマイクロプロセッサを組み入れることは現在では当り前になりました．マイクロプロセッサ（またはそれを含む“マイコン”）とは，それほど便利で汎用性のある部品です．しかし，その便利さを発揮するためには，組み込まれる装置ごとにインターフェースを設計しなければなりません．

一方，市販のパーソナル・コンピュータの価格が下がり，マイコンは作る時代から買う時代に変わりました．このため，誰でも簡単にマイコンの恩恵にあずかれるようになりましたが，ちょっとしたインターフェース回路ですら自分では作れない人も増加しています．

とかく，マイコンを使いこなすこととはソフトウェアが作れることであると思われがちです．しかし，より広範囲の応用を考えるならば，ハードウェアの知識も必要です．

本書は，マイコンのソフトウェアには触れたことがあるがハードウェアには自信がないという初心者の方々に，設計のヒントとしていただこうと意図したものです．また，より上級の方々にも基礎知識の復習となれば，幸いです．

なお，マイコンを使って何かを作り上げることは，きわめて総合的かつ創造的な仕事であり，本書一冊ではとうてい全部をカバーすることはできません．このため，ソフトウェアをはじめとして本書に書かれていない部分は他の書籍等を参考にしていただくとともに，お気付きの点は遠慮なくご指摘下さい．

1985年　春　　　著者

目　次

編集担当／山本　潔
〈カバー〉
デザイン／宰　良二
フォト／佐瀬　真

第1章

マイコン・システムの構成法

マイコンの応用範囲はますます広がりつつあります．以前はマイコンそれ自体が完結したシステムと考えられていたのに対して，最近の使われ方を見ると，マイコンは汎用コントローラとして大きなシステムの一部に組み入れられることが多くなりました．

しかも，マイコンの守備範囲はきわめて広く，システムの分散処理指向とあいまって，多品種少量生産品のコントローラとしては欠かせないものとなっています．

もちろん，これはマイコンの汎用性——ソフトウェアと一部分のハードウェアとを取り換えれば“何でも”できる点——を買われてのことです．しかし，その利点を活かすためには，似てはいても少しずつ異なるソフトウェアとハードウェアとを，そのシステムごとに設計しなければなりません．

このわずらわしさは，とくにソフトウェアについて，最近やかましくいわれるようになってきました．ハードウェアに関しては従来はさほど問題とされていませんでしたが，メーカ製のマイコンの価格が量産効果によって驚くほど下がり，その中身の詳細を知らずに使う時代となってきた結果，逆に新たな問題となっているようです．つまり，ごく簡単なインターフェースを製作するにも，必要な情報が公開されていなかったり，担当者が細部にわたる知識をもっていなかったりするという理由で最適な設計が行えない事態が少なからず存在するのです．

最適でなくとも確実に動作すれば良いのですが，しばしば頭を痛めるのは，既製のマイコンと外部との間で簡単なやりとり（たとえばスイッチを接続したり，ランプを点灯したりという程度の）を行うためのインターフェースを追加したところ，“ときどき”マイコンがエラーを起こす，という症状がでることです．

もちろん，追加したインターフェース回路はちゃんとした書籍や雑誌に発表されたもの

であり，単体で機能チェックを行うと，完全に動作しています．マイコンのほうも単体ではエラーなど生じません．では，このような事態がなぜ生じるのか，また正しく設計するにはどうすればよいのか，それらの点を説明することを本書の目的とします．

1.1　マイコンを使ったシステムの形態

マイコンといっても，1個のIC（シングルチップ・コンピュータ）に電源を接続しただけのものもあれば，CRT（ブラウン管）やキーボードのついたパーソナル・コンピュータもあります．また，その用途もいろいろです．

図1.1に示すのは，マイコンを使った種々のシステム例です．同図(a)は1枚の基板上にCPUやインターフェース回路を作り，組込み型のコントローラとしたものです．以前なら

図1.1 (a)　マイコン・システムの形態例(専用の制御基板)

(a) 専用の制御基板

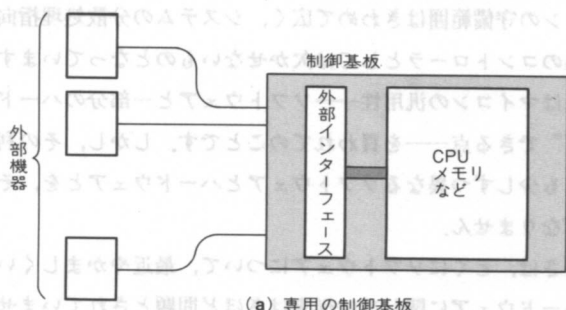



図1.1 (b)　マイコン・システムの形態例(専用システム)

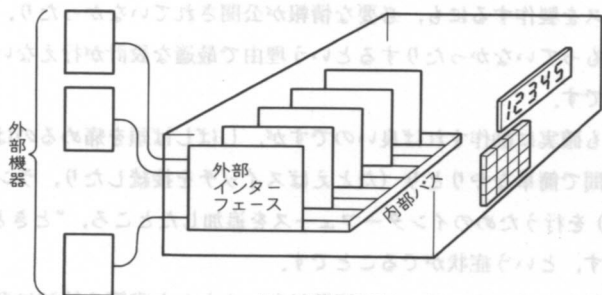
(b) 専用システム

ばハードウェアで組んでいた制御回路も，最近はこのようにマイコンを使ったものが増えてきました．このようなシステムでは，仕様の範囲を正確に見定めて，必要な機能をコンパクトにまとめることが重要です．

図1.1(b)に示すのはやや規模が大きい場合で，マイコンを使ったコントローラを別の箱に入れて各種の制御を行わせようとするものです．このような構成では，回路規模を小さ

図1.1 (c) マイコン・システムの形態例(パーソナル・コンピュータを用いた構成)

(c) パーソナル・コンピュータを用いた構成

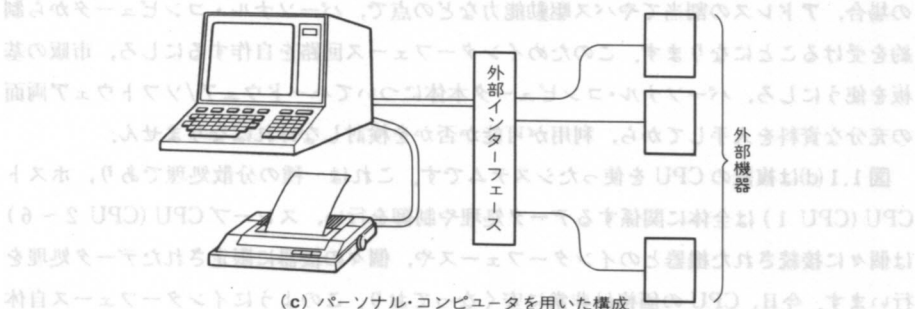



図1.1 (d) マイコン・システムの形態例(複数CPUシステム)

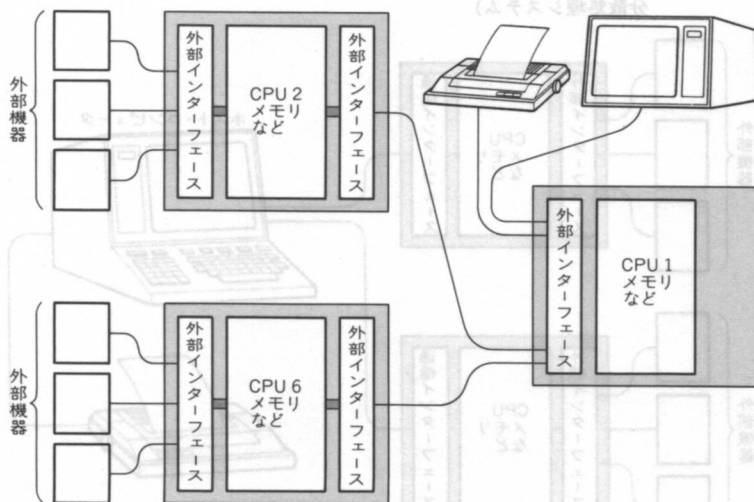
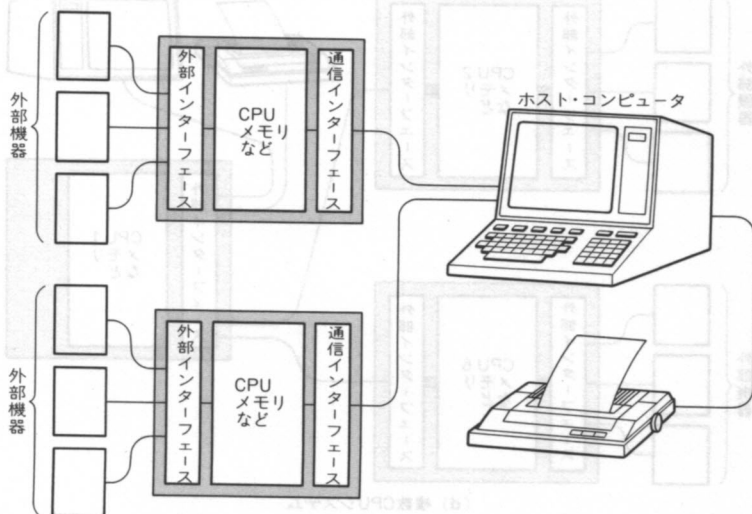
(d) 複数CPUシステム

くすることよりも，専用コントローラとしての機能を充実させることが重要です．

図1.1(c)は市販のパーソナル・コンピュータにインターフェース部分を増設したものです．パーソナル・コンピュータにはCPUなどマイコンとしての中核部分のほか，CRT表示器やキーボードなどの入出力装置，直列通信回線（RS-232Cなど）やプリンタ・インターフェースなど，汎用的な入出力インターフェースがすでに付属しています．したがって，これらの機能の一部を使うつもりならば，安価で便利なシステムとなります．ただし，ごくかぎられた用途を除いて，入出力インターフェース回路を追加する必要があります．その場合，アドレスの割当てやバス駆動能力などの点で，パーソナル・コンピュータから制約を受けることになります．このためインターフェース回路を自作するにしろ，市販の基板を使うにしろ，パーソナル・コンピュータ本体についてハードウェア/ソフトウェア両面の充分な資料を入手してから，利用が可能か否かを検討しなければなりません．

図1.1(d)は複数のCPUを使ったシステムです．これは一種の分散処理であり，ホストCPU（CPU 1）は全体に関係するデータ処理や制御を行い，スレーブCPU（CPU 2～6）は個々に接続された機器とのインターフェースや，個々の機器に限定されたデータ処理を行います．今日，CPUの価格は非常に安くなっており，このようにインターフェース自体

図1.1（e）　マイコン・システムの形態例（パーソナル・コンピュータを中心とした分散処理システム）

（e）パーソナル・コンピュータを中心とした分散処理システム

をマイコンで作ることも増えてきました．このような構成では処理速度も上がり，ソフトウェアも別々に作れば良いので，むしろ簡単になるでしょう．また，同図(e)のように中心にホスト・コンピュータ（パーソナル・コンピュータ）を置けば，パーソナル・コンピュータと専用機との長所を補い合ったシステムが作れます．このようなシステムでは，各CPU間の仕事の分担のしかたと，CPU間の連絡方法(やはりインターフェース)とが設計のポイントとなります．

1.2 マイコンを使ったシステムの設計手順

以上のように，マイコンを使ったシステムには様々な形態があります．また，接続したい対象も非常に多く，いざ設計しようとしても何から手を付けたら良いかわからないこともあると思います．

そこで，設計の際に最小限必要なステップとして表1.1のような順序を考えてみます．もちろん，設計はこの順序で行わなければならないものではありませんが，このようなシーケンスが頭に入っていれば設計に必ず役立ちます．

●必要とされる入出力を見極める

まず，システム全体を頭に入れて，どのような入出力が必要かを考えます．入力とは電気信号で外部から与えられるもののほか，各種のセンサやトランスジューサのこともあります．また，人間が操作するものとそうでないものとがあります．

一方，出力も電気的なものだけではなく，音や光による合図，プリンタやCRTへの表示なども忘れないように拾い上げていきます．

一般に，マイコンに接続されるものの1例を図1.2に示します．結論からいえば，マイ

表1.1 入出力インターフェース設計の手順の例

(1) どのような入出力が必要か．
(2) 各入出力について，どのような機能(仕様)が必要か
- ●使用目的と使用方法．
- ●形態やレベル
- ●雑音対策，伝達距離，インピーダンス
- ●マイコン側から見た速度

(3) ソフトウェア/ハードウェアの分担を決める．
(4) ソフトウェア/ハードウェアの設計
(5) デバッグ

コンには“何でも接続したい”という要求があり，この図に示したものも1例に過ぎません．むしろ，この図に出ていないものを発見し，独創的なインターフェースを設計してマイコンに接続することこそ，本当にマイコンを活用することにつながります．

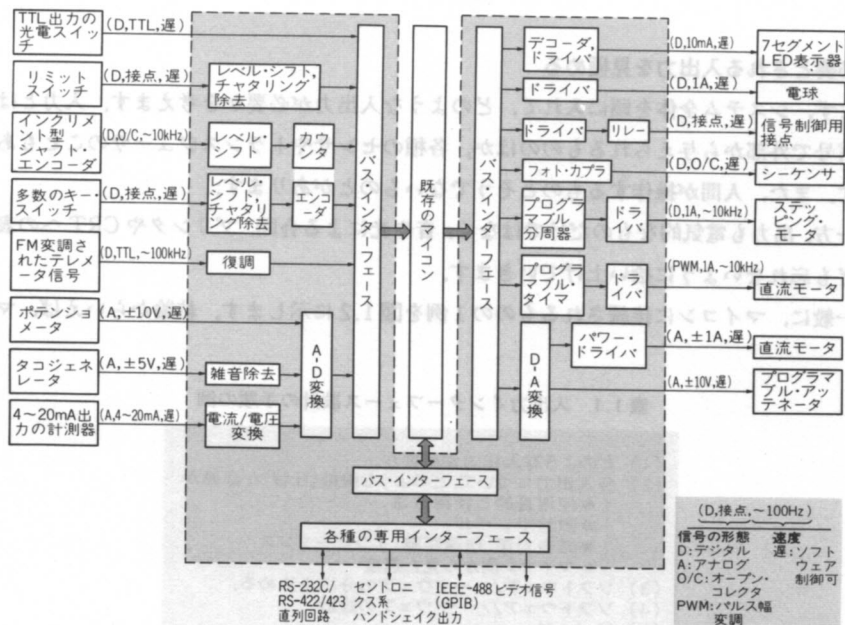
●入出力の仕様を定める

つぎに，それぞれの入出力について具体的な内容を決めます．この作業はきわめて重要で，インターフェース回路の設計の可否はこの段階で左右されます．

では，入出力の仕様として何に着目すれば良いのでしょうか．

(1) **信号の使用目的と使用方法**　まず何のために必要な信号かを良く考えて下さい．後で良く考えたら必要な信号ではなかったとか，逆に信号が不足したというのは，この段階での失敗です．

次に重要なのは信号の使用方法です．たとえば，人間（操作者）が操作するスタート・スイッチにしても，

図1.2　マイコンに何をつなぐか

(i) 押しボタン・スイッチ（モーメンタリ）にして1回押せばスタートする．

(ii) 押しボタン・スイッチ（モーメンタリ）だが押している間だけ動く．

(iii) トグル・スイッチにしてパチンと切り換えてしまう．

など，いろいろな操作方法があります．もっとも操作しやすく，間違いが少ないものを選ぶのが重要で，このことがマン・マシン・インターフェースの第一歩につながります．

一方，機械と機械との信号のやりとりにも同じことがいえます．たとえば動作中か否かを別の機器に知らせる電気信号でも，動作中であることをレベルで示すのと，動作終了時にパルスを出すのとでは，それぞれ長所，短所があります（詳細は第3章を参照）．

(2) **信号の形態やレベル**　つぎに可能なものについては電気的な仕様を決めていきます．信号の形態やレベルとは，それがアナログかディジタルか，接点か電圧か，その電圧はどれくらいか，などを指します．

(3) **雑音，信号の伝達距離，インピーダンス**　外部機器側の信号として高電圧や大電流を扱う場合，これがマイコン本体や他のインターフェースに及ぼす影響を考慮します．また逆に，微小信号を扱う場合は，ほかからの雑音混入に注意します．特に，マイコン自体は大きなディジタル回路であり，周囲に少なからぬ雑音を与えることがあります．

一方，長い信号線を必要とする場合や，信号源インピーダンスが高い場合，信号伝達に注意が必要となります．

(4) **必要とされる速度**　これは，ハードウェアとソフトウェアとの分担を決める上で非常に重要ですから，慎重に検討します．

速度には，つぎの2種類があります．

(i) 変化の周期

(ii) 必要とされるタイミングとのズレ

変化の周期は入出力のサンプリング間隔を選ぶのに重要です．ソフトウェアでループを作った場合，それを1回まわるには簡単なものでも数10～100 μs が必要です．したがって，入力であっても出力であっても，ソフトウェアを用いて100 μs 以下の信号変化周期を扱うのは困難です．

つぎに，必要とされるタイミングとのズレについて考えます．たとえば，変化周期は遅いものの，ストローブ信号が印加されてから一定時間以内にデータを入力，あるいは出力しなければならないことがあります．また，複数の入力信号が同時に変化したか否かを調べたり，複数の出力信号を同時に変化させたい場合もあります．

ソフトウェアは必ずしも“柔軟”ではありません．たとえば別々のアドレスに割り当て

られた信号を同時に変化させることは，ハードウェアで手段を設けていないかぎり不可能です．したがって，この問題はハード/ソフトのトレード・オフとも関連して検討しておかなければなりません．

●ソフトウェア/ハードウェアの分担を決める

すでにおわかりのように，マイコンはソフトウェアとハードウェアとの適切な組合せがあって初めて所定の動作が可能となります．

しかし，その場合，ソフトとハードとの分担を決めることが必要となります．これは仕事の種類によってつぎの3種類に分けられます（図1.3）．

(1) ハードウェアでは可能だが，ソフトウェアでは不可能またはきわめて困難なもの．

(2) ハードウェア，ソフトウェアのいずれによっても可能なもの．

(3) ソフトウェアでは可能だが，ハードウェアではきわめて困難なもの．

まず(1)に属する仕事に対しては必ずハードウェアを設けなければなりませんので，この範囲を定めることはきわめて重要です．

(1) ハードウェアでは可能だが，ソフトウェアでは困難なもの　まず，仕事の内容を2つに分けて考えてみます．その第1は信号の形態や，他の電気的仕様の変換です．これは，ほとんどの場合にハードウェアの仕事になるでしょう．たとえば，CPUに直接アナログ信号を接続しても，ソフトウェアではどうしようもありません（図1.4）．

第2の仕事は，データの転送や変換などの処理です．これについては現在のマイコンと，それに対する要求とから見て，ソフトウェアでもっとも苦手なのは第1に高速性を要求さ

図1.3 ハードウェア/ソフトウェアの分担

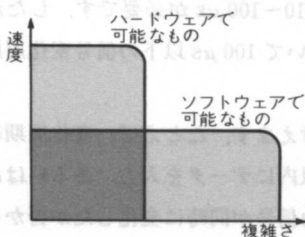
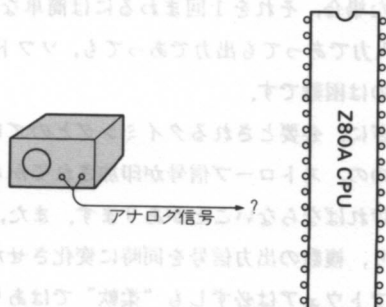



図1.4 ディジタル用の端子にアナログ信号はつなげない

れるもの，第2に多数の事象の並列処理であるといえます．

まず，前に述べたように，一定以上の速度が必要とされる場合はソフトウェアでは対応できません．

また，複数の事象の並列処理もソフトウェアでは困難です．簡単なものでは多数の入力信号が“同時”に変化したか否かを調べたり，多数の出力信号を“同時”に変化させることは，ソフトウェアだけでは困難です．これは，現在のマイコンが一連の命令列を順に実行する（いわゆるノイマン型）かぎり，別々のアドレスに割り当てられた信号を厳密に同時に扱うことは不可能だからです．したがって，このような仕事もハードウェアのお世話になります．

(2) **ハードウェア/ソフトウェアのいずれでも可能なもの**　これは“簡単な内容であって速度も遅くてよいもの”になります．ただし，ハード，ソフトの両方で可能であっても，それぞれに特徴があることを忘れてはなりません．

たとえば，スイッチのチャタリング除去（第3章参照）をハード，ソフトのそれぞれで行った場合を考えてみます（図1.5参照）．一般にソフトの利点といわれているのは，

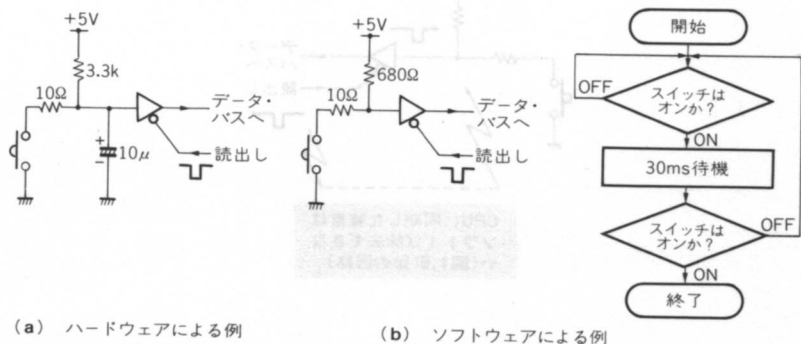
(i) 部品数が減る

(ii) 時定数を簡単に（ハンダなしに）変更できる

という点です．このほかに，部品の経時変化の影響を受けにくい点も見逃せません．

しかしながら，図1.5(a)の回路は図1.6のようなパルス性雑音の除去もできますが，ソフトウェアではできないことがあります（たとえば，マイコンの動作自体に同期した雑音は除去できない）．また，実際に稼動し始めたあとのソフトウェアを変更するのは困難な場

図1.5　チャタリング除去

(a) ハードウェアによる例　　(b) ソフトウェアによる例

合も多く，上記(ii)は必ずしもソフトの利点とはいえません．

このように，ハード，ソフト両方で可能なものでも両者の特徴をよく整理して選択する必要があります．

(3) ハードウェアでは困難なもの　ハードウェアで困難な仕事とは，複雑な処理や演算を必要とするもののことです．普通はこのような仕事はソフトで行いますので，本書では触れません．ただし，経済性などを無視すればソフトで実現できるものは原則としてハードに置き換えることが可能（逆は正しくない）だという点は覚えておく必要があります．

●ソフトウェア/ハードウェアの設計

入出力やソフト/ハード分担の仕様が決まったら，いよいよ設計に入ります．このうち，入出力インターフェースのハードウェアについては本書で詳しく説明します．ソフトウェアは他の参考書等をご覧下さい．その場合に注意すべきなのは次のようなことです．

(i)　常にシステム全体への要求を頭に入れておくこと．

仕様の１つ１つを満足することをなおざりにしてはいけません．しかし，それにはシステム全体が働くことが大前提となります．

(ii)　入手可能な情報をフルに活用すること．

たとえば市販のパーソナル・コンピュータにインターフェースを追加する場合，少なくともその中身について知っておく必要があります．ソフトウェアを設計するにも，使用するCPUの動作は知っておくべきです．

図1.6　ソフトウェアでは除去できない雑音

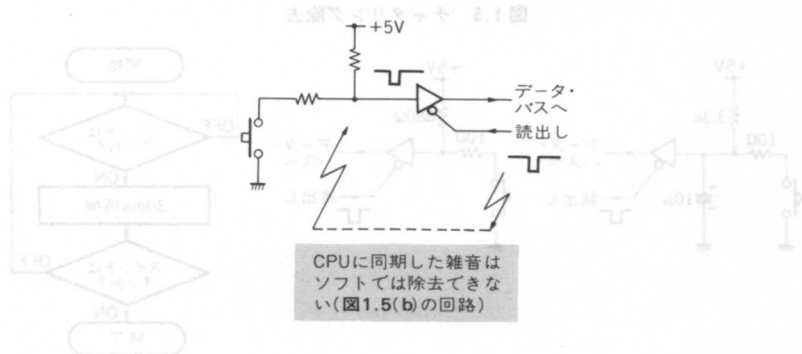
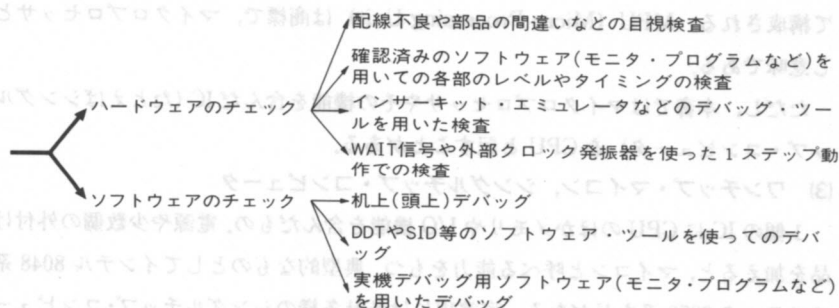



表1.2 デバッグの方法──その1例

●ソフトウェア/ハードウェアのデバッグ

マイコンを使ったシステムでもっとも困るのがデバッグです．へたをすると，原因がハードかソフトかわからないまま，全く動かないシステムができてしまいます．

デバッグの方法そのものは本書の目的ではありませんが，デバッグを考慮したインターフェースを設計することは重要です．とくに，組込み型のコントローラでは，最初から実機デバッグを考慮しておかないかぎり，動かなかったりしたときに手のつけようがありません．これには表1.2に示すような方法があるので参考にして下さい．

コラムA　マイコンとは

世の中にはマイコン，パソコン，マイクロプロセッサなど似たような名称が多く存在します．この種の単語は，定義など関係ないと無意識の選択によって使われることもあれば，意識的に異なった意味で区別して使われることもあり，事態をより複雑にしています．そこで，本書ではつぎのような意味で使うことを決めておきます．

(1)　マイコン（マイクロコンピュータ）

単体で動作し，CPU(マイクロプロセッサ)，メモリ，入出力手段などの基本的な要素を含むもの．たとえば，市販のパーソナル・コンピュータも，組込み用のCPU基板に電源を付けたものも，マイコンの1つである．本書では“マイコン”という語が数多く登場するが，このように広い意味で使用している点に留意してほしい．

(2)　CPU，マイクロプロセッサ，MPU

CPUは計算機の中央処理装置（Central Processing Unit）のことだが，それをLSI

化したものがマイクロプロセッサ．マイコンは通常，マイクロプロセッサを中心として構成される．MPU（Micro Processing Unit）は商標で，マイクロプロセッサと同じ意味である．

ただし，本書ではマイクロプロセッサやその機能を含んだIC（たとえばシングルチップ・コンピュータ）をCPUと記すことがある．

(3)　ワンチップ・マイコン，シングルチップ・コンピュータ

1個のICにCPUのほかメモリやI/O機能を含んだもの．電源や少数個の外付け部品を加えると，マイコンと呼べる能力をもつ．典型的なものとしてインテル8048系やモステック3870系などがある．家電用には各社各様のシングルチップ・コンピュータが多く使われている．

第2章

マイコンの構造と動作

マイコンのインターフェースを設計するには，マイコンの構造や動作を知っておく必要があります．

2.1　マイコンの動作

そこで，簡単な例をあげて，実際のコンピュータの動作を調べてみましょう．

図2.1(a)に示すのは，Z80 CPUを用いた簡単なプログラムです．これは$8000_{(16)}$番地の内容に$30_{(16)}$を加えて$8001_{(16)}$番地に格納するもので，8000番地を入力ポート，8001番地を出力ポートとすれば，概念的には図2.1(b)のようなハードウェアと同じ働きをします．

しかし，図2.1(a)と(b)との働きは良く似ていても，実際の動き方は全く異なります．図

図2.1　簡単なプログラムと等価ハードウェア

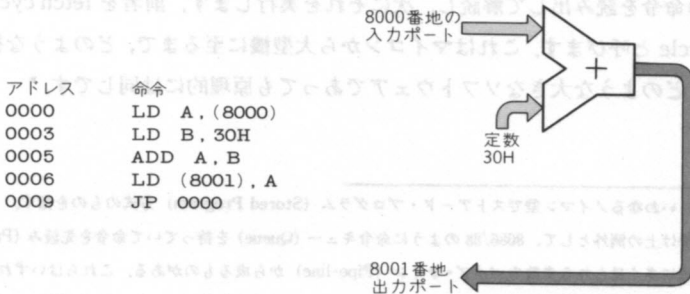
アドレス	命令
0000	LD　A，(8000)
0003	LD　B，30H
0005	ADD　A，B
0006	LD　(8001)，A
0009	JP　0000

(a)　簡単なソフトウェア　　(b)　等価ハードウェア

図 2.2　マシン・サイクルごとの動作

ニモニック	機械語	マシンサイクルごとのバスの動作			
LD　A,(8000H)	3A	M1	0000番地からの読出し	OPコードの解読	命令読出し
	00	M2	0001番地からの読出し	アドレスの読出し	↓
	80	M3	0002番地からの読出し	アドレスの読出し	
	—	M4	8000番地からの読出し	…………………	命令実行
LD　B,30H	06	M1	0003番地からの読出し	OPコードの解読	命令読出し ↓
	30	M2	0004番地からの読出し	パラメータの読出し	命令実行*
ADD　A,B	80	M1	0005番地からの読出し	OPコードの解読→命令読出し→命令実行*	
LD　(8001H),A	32	M1	0006番地からの読出し	OPコードの解読	命令読出し
	01	M2	0007番地からの読出し	アドレスの読出し	
	80	M3	0008番地からの読出し	アドレスの読出し	
	—	M4	8001番地への書込み	…………………	命令実行
JP　0000H	C3	M1	0009番地からの読出し	OPコードの解読	命令読出し
	00	M2	000A番地からの読出し	アドレスの読出し	↓
	00	M3	000B番地からの読出し	アドレスの読出し	命令実行*

＊の命令実行にはバス動作を行わない

2.2に示すのは，Z80 CPUのマシン・サイクルを単位として，図2.1(a)のプログラムの実際の動きを示すものです．このように，CPUは命令読出しと命令実行とを交互に繰り返します．また，命令読出しの際には必ずメモリからの読出しを行い，命令実行の際には必要に応じて読出しまたは書込みを行います．

●命令読出し/命令実行

一般に，現在稼動している大半のコンピュータは，まずメモリ（プログラム・メモリ）から1つの命令を読み出して解読し，次にそれを実行します．前者をfetch cycle，後者をexecute cycleと呼びます．これはマイコンから大型機に至るまで，どのような複雑なハードウェア，どのような大きなソフトウェアであっても原理的には同じです．*

---

*　ここでは，いわゆるノイマン型でストアード・プログラム（Stored Program）方式のものを指す．

ただし，見掛け上の例外として，8086/88のように命令キュー（Queue）を持っていて命令を先読み（Pre fetch）するものや，大型機に多く見られる多数のパイプ・ライン（Pipe-line）から成るものがある．これらはいずれも，命令読出しと命令実行とを順に繰り返しているようには見えない．しかし，前者の実行ユニット，後者の個々のパイプ・ラインを各々注目すれば，やはり命令読出しと命令実行とを順次繰り返しているに過ぎない．

図 2.3 コンピュータの構造を示す図

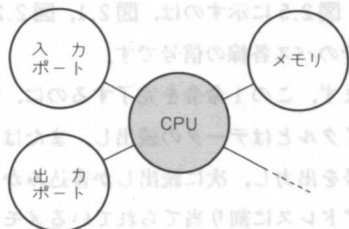
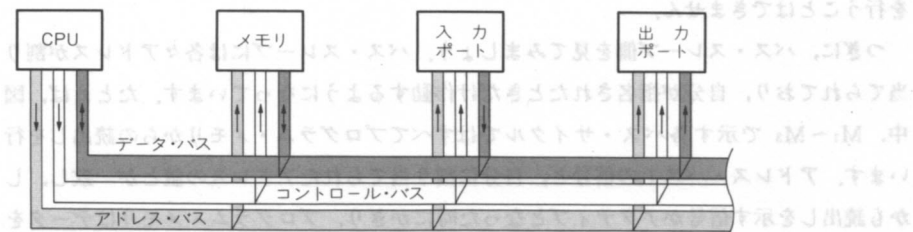



図 2.4 共通バス構造のコンピュータの構造概念図

●バス構成

以上のように，CPU は命令読出しや実行に際して外部のメモリ，入力ポート，出力ポートとの間でデータのやりとりを行います．一般に，CPU が外部に対して行う仕事のほとんどすべてが，このデータのやりとりに関するものだといっても過言ではありません．*では，データのやりとりは具体的にどのように行われるのでしょうか．

図 2.3 は，コンピュータの構造を示すのに良く使われる図です．確かに，このように各要素に専用のデータ線を持ったコンピュータもあります．しかし，とくにマイコンやミニコンに類するものでは，共通バス（Common bus：共通母線）構成となっているのが普通です．

図 2.4 は共通バス構成のコンピュータの構造概念図です．このように，どの要素もバスに並列に接続されています．また，アドレス・バスやコントロール・バスは信号の方向が変化しない（常に，CPU から出力される）のに対して，データ・バスは読出し時と書込み時とで信号の方向が異なる点に注目して下さい．

●バスの動作

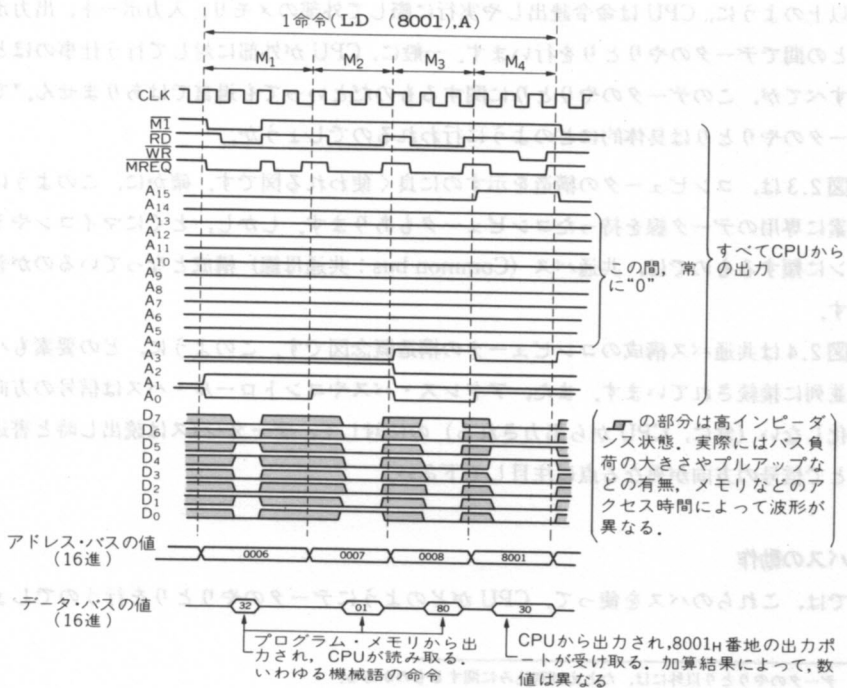
では，これらのバスを使って，CPU がどのようにデータのやりとりを行うのでしょう

* データのやりとり以外には，たとえば割込みに関するものがある．

か．図2.5に示すのは，図2.1，図2.2のプログラムのうち，LD　(8001), A に相当する部分のバス各線の信号です．

まず，この1命令を完了するのに，CPUは4回のバス・サイクルを実行します．バス・サイクルとはデータの読出し，または書込みのことです．各バス・サイクルではアドレス信号を出力し，次に読出しか書込みかを示す信号を出力します．したがって，普通は別々のアドレスに割り当てられているメモリや入出力ポートに対して，同時に読出しや書込みを行うことはできません．

つぎに，バス・スレーブ側を見てみましょう．バス・スレーブには各々アドレスが割り当てられており，自分が指名されたときだけ作動するようになっています．たとえば，図中，M_1～M_3 で示す各バス・サイクルではすべてプログラム・メモリからの読出しを行います．アドレス・バス上の信号と，自分に割り当てられたアドレスの値とが一致し，しかも読出しを示す信号がアクティブとなった時にかぎり，プログラム・メモリはデータを

図2.5　LD　(8001), A に相当するバス各線の信号

データ・バスに出力します．これ以外の場合は，データ・バスの衝突の可能性があるので，データ・バスにデータを出力してはなりません．

一方，バス・サイクル M_4 では，CPU から出力ポートへの書込みを行います．書き込まれる側のバス・スレーブ（この場合は 8001H 番地の出力ポート）は，自分が指名されるとともに書込み信号が印加された場合にかぎり，データ・バス上の値を固定します．出力ポートは単なるレジスタで，最も簡単には D フリップフロップを考えれば良いでしょう．

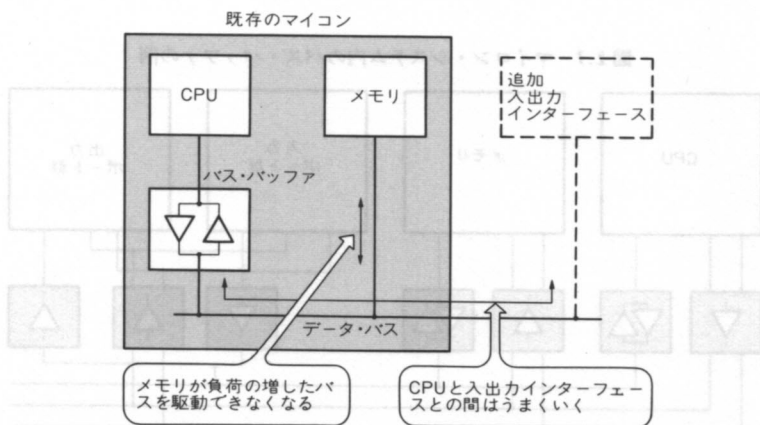
2.2 マイコン・システムにおけるバス

以上でおわかりのように，CPU，メモリ，入出力ポートなど，マイコンの主要な要素はすべてバスに接続されます．マイコン・インターフェースの設計には，バスを正しく利用することが重要です．では，バスを利用するのに必要な事項を検討してみます．

●バス負荷の考え方

バスにはマイコンのほとんどの要素がつながります．とくに，データ・バスには入力，出力の両方の要素が接続されています．

しかし，バスに対して信号を出力できるのは，一度に1つの要素だけにかぎられます．そしてこのとき，バスに出力した信号が，他の要素に正しく受け取られることが必要です．

図 2.6　動きそうで動かない例

そのためには，バスに信号を出力する要素が，バスにつながった他の要素を確実に駆動できなければなりません．

よくある設計ミスを図2.6に示します．追加したインターフェースと実際にデータをやりとりするのはCPUだけです．しかし，それ以外の要素にとっても追加インターフェースは負荷となることを忘れてはなりません．

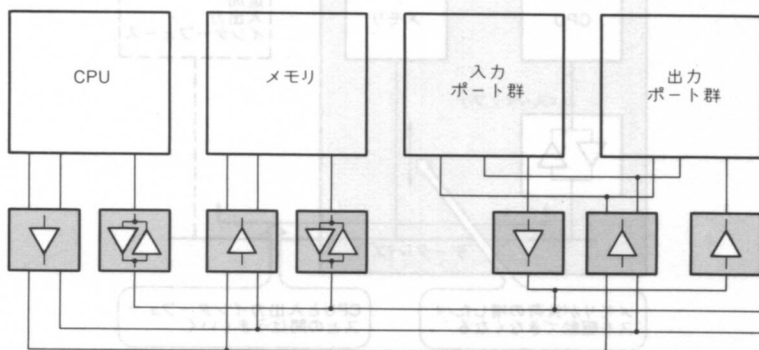
●バス・バッファ

比較的小さなシステムでは，CPUやメモリは直接，バスに接続できます．しかし，バス負荷が増加すると，それを直接には駆動できなくなるのでバッファが必要となります．

図2.7に示すのはバッファの使い方の例です．この例ではCPUやメモリのデータ・バスは双方向性なので，双方向バッファを使っています．一方，入力ポート，出力ポートは別々に一方向のバッファとしています．このように，部分的に一方向のバスを使うほうが回路が簡単になる場合があります．

ちなみに，Z80A CPU（NMOSのもの）では，直流的にはLS-TTL（普通のもの）ならば4個まで駆動できます．一方，バス・タイミングは容量負荷50pFで規定されていますので，MOSメモリのように直流負荷が事実上無視できるものであっても，5個くらいまでにしたほうが良いでしょう．つまり，それ以上の要素を接続する場合はバッファを必要とします．

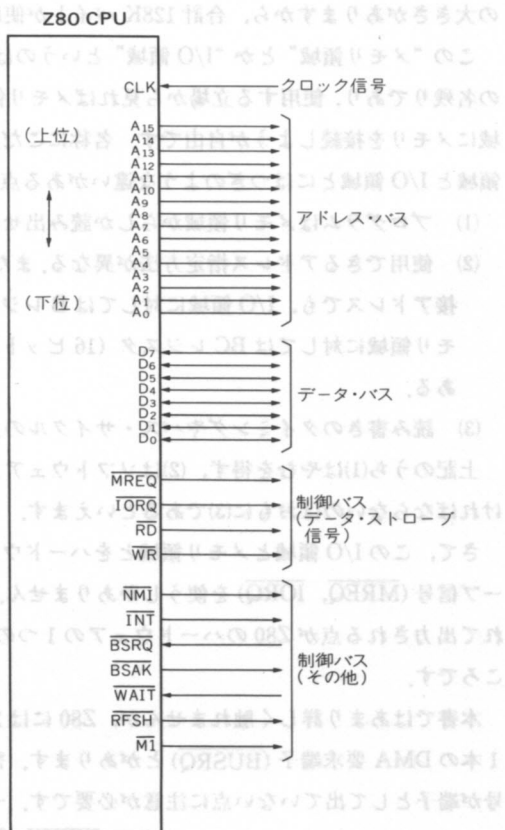
マイコンで何かをしようとしたとき，ソフトウェア，ハードウェアのいずれか一方だけ

図2.7　マイコン・システム内のバス・バッファの例

では満足なシステムが作れないことがおわかりいただけたと思います.

ハード/ソフトの両面を考えたシステム設計を行ってこそ,マイコンのもつ長所が有効に活用できるのです.

2.3 Z80 CPUバス[3]

世の中ははやくも32ビットの時代に突入しようとしているのですが,Z80は値段の安さとソフトウェア・ツールの豊富さから未だにコントローラ用として使う機会が少なくありません.

図 2.8 Z80 CPUのバス構成

もちろん，性能上の批判が何かと多いことも，そのバス構成が決して一般的ではなく本項のような基礎的解説に向いていないことも承知のうえで，あえて現場的見地からここではZ80をモデルに話をすすめることにします．

なお，他のCPUについては，次節でその特徴を要約してあります．もちろん，これ以外にもCPUの種類は数多く，筆者としては個人的には捨て難いものもあるのですが，今回は広く知られていないものは割愛させていただきました．

●Z80のバス構成

Z80は8ビットのデータ・バスと16ビットのアドレス・バスをもつCPUです（図2.8）．アドレス空間にはメモリ領域とI/O領域とがあり，それぞれ16ビット（64Kバイト）の大きさがありますから，合計128Kバイトが使用できます．

この“メモリ領域”とか“I/O領域”というのはZ80の前身8080から受け継いだソフトの名残りであり，使用する立場から見ればメモリ領域にI/Oポートを接続しようがI/O領域にメモリを接続しようが自由です．名称にこだわることはありません．ただし，メモリ領域とI/O領域とにはつぎのような違いがある点に注意してください．

(1)　プログラムはメモリ領域からしか読み出せない．

(2)　使用できるアドレス指定方法が異なる．また，両者ともに使用できるBCレジスタ間接アドレスでも，I/O領域に対してはBレジスタ（8ビット）のインクリメント，メモリ領域に対してはBCレジスタ（16ビット）のインクリメントを行うなどの違いがある．

(3)　読み書きのタイミングやバス・サイクルの長さが異なる（I/O領域のほうが遅い）．

上記のうち(1)はやむを得ず，(2)はソフトウェア上の問題で，ハード設計上知っておかなければならないのはおもに(3)であるといえます．

さて，このI/O領域とメモリ領域とをハードウェア的に区別するには，データ・ストローブ信号（$\overline{\mathrm{MREQ}}$，$\overline{\mathrm{IORQ}}$）を使うしかありません．これらの信号がアドレスからかなり遅れて出力される点がZ80のハードウェアの1つの欠点であることはよく知られているところです．

本書ではあまり詳しく触れませんが，Z80には2本の割込み要求端子（$\overline{\mathrm{NMI}}$，$\overline{\mathrm{INT}}$）と1本のDMA要求端子（$\overline{\mathrm{BUSRQ}}$）とがあります．割込みに関しては割込みアクノリッジ信号が端子として出ていない点に注意が必要です．一方，DMAはデータ・バス，アドレス・バスのほか，データ・ストローブ信号（$\overline{\mathrm{MREQ}}$，$\overline{\mathrm{IORQ}}$，$\overline{\mathrm{RD}}$，$\overline{\mathrm{WR}}$）がすべて高インピー

ダンスとなります．

●バス・サイクルの種類

Z80は実行する命令により，必要とするマシン・サイクルの数が1～5に変化し，またバス・サイクルも5種類あります．

Z80の命令実行の例を表2.1に示します．

まず，どのような命令であっても，その第1バイト目（OPコード）を読み出すには専用のバス・サイクル（M1サイクルと呼ぶ）を用います．これは，命令の種類を少しでも早くデコードして実行に移すためと，ダイナミックRAMのリフレッシュをこのサイクルで行うためのものです．

命令の第2バイト目以降の読出しとメモリ領域からのデータ読出しとには，メモリ読出しサイクルを用います．一方，I/O領域からのデータ読出しにはI/O読出しサイクルを用います．I/O領域のバス・サイクルの長さをわざわざ変えてあるのは，8080A時代の遅い周辺ICの使用を考えたためです．同様に書込みの際にも，メモリ書込みサイクルとI/O書込みサイクルとの区別があります．

表2.1　Z80の命令実行の様子

命令の例		LD A, B	LD A, n	LD (nn), BC	IN A, (BC)
クロック数とマシン・サイクル	1	M1 OPコードの読出し	M1 OPコードの読出し	M1 OPコードの読出し	M1 OPコードの読出し
	2				
	3				
	4				
	5		MR nの読出し	MR 命令2バイト目の読出し	MR 命令2バイト目の読出し
	6				
	7				
	8			MR nnの読出し	IOR 読出し
	9				
	10				
	11			MR	
	12				
	13				
	14			MW 書込み	
	15				
	16				
	17			MW	
	18				
	19				
	20				

図 2.9　Z80A CPU のメモリ読出しサイクル

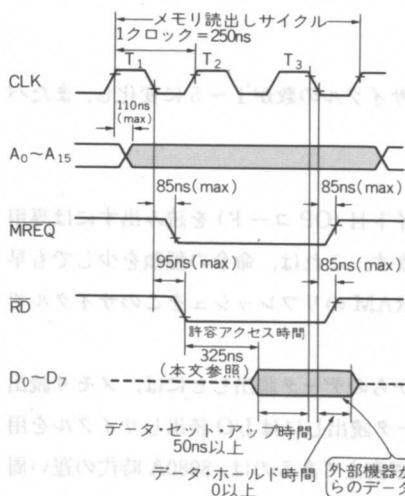
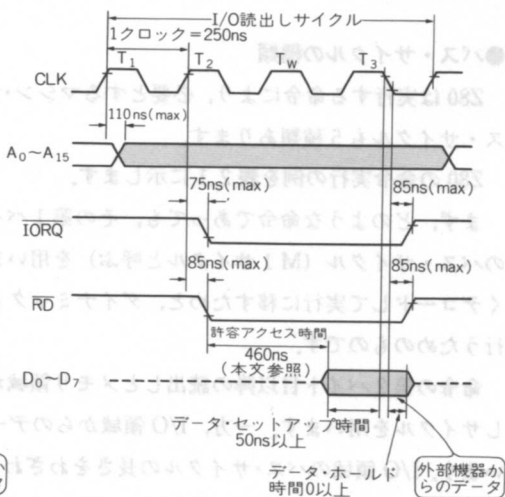



図 2.10　Z80A CPU の I/O 読出しサイクル

では，つぎにこの5種類のバス・サイクルのタイミングを以下に説明します．

(1)　**メモリ読出しサイクル，I/O 読出しサイクル**　図2.9にメモリ読出しサイクルのタイミングを示します．なお，ここで説明するのはZ80A をクロック周波数 4 MHz で使用した場合の数値です．

図のいちばん上に書いてあるのはCPU に印加するクロック信号です．普通，入出力インターフェースの設計にはクロック信号を直接使用することはない*でしょうが，CPU の内部はこのクロックに同期して動いており，タイミング計算上もクロックが基準になります．

メモリ読出しサイクルはクロック信号の立上りから始まります．これに同期して，アドレスも出力されます(ただし，CPU 内部の遅れ時間だけ遅れて確定する)．つぎに，クロックの半周期後に $\overline{\text{MERQ}}$ と $\overline{\text{RD}}$ とが“L”になります．

一般に，入力用インターフェースはこの時点で必要とするすべての情報（アドレスが所定の値か？　指定されたのはメモリ領域か I/O 領域か？　読出しか書込みか？）がそろうことになり，必要とあらばデータの出力を開始します．

CPU が内部にデータを取り込むのは3個目のクロック（T_3）の立下りです．したがっ

*　Z80 ファミリの周辺 IC は CPU と同じクロック信号を必要とする．

て，すべての情報がそろってからデータを確立させるまでに2クロック分の時間的余裕があることになります．ただし，$\overline{\text{MERQ}}$や$\overline{\text{IORQ}}$が出力されるのに要するCPU内部の遅れ時間や，CPUのデータ取込みに必要なデータ・セットアップ時間を差し引く必要があります．

本書ではこの時間（データ・ストローブ・アクセス時間）をたんにアクセス時間と呼ぶことにします.* 図2.9の場合，入力インターフェースに許されるアクセス時間の最大値は，クロックの立上り・立下り時間を各30nsと仮定すれば325nsとなります．入力インターフェース回路はこの時間内にアドレスおよびデータ・ストローブ信号をデコードし，スリーステート（3ステート）・バッファの出力をアクティブにしなければなりません．

一方，図2.10に示すのはI/O読出しサイクルです．メモリ読出しサイクルと異なるのは，バス・サイクル自体が1クロック分だけ長いこと，$\overline{\text{IORQ}}$および$\overline{\text{RD}}$が2個目のクロック（T_2）の立上りで変化することです．この結果，アクセス時間の許容値は約半クロック分だけ延びて460nsとなります．

なお，Z80 CPUはデータの読み書きが間に合わない場合，WAIT端子を用いてバス・サイクルを延長することができます．しかしTTLを組み合わせてつくった入出力インターフェースでCPUを待たせなければならないことはまれです．むしろ，CPUを待たせないように設計することに留意すべきだと思われます．

⑵　**メモリ書込みサイクルとI/O書込みサイクル**　　図2.11に示すのがメモリ書込みサイクルのタイム・チャートです．

読出しの場合と同様に，アドレスはバス・サイクルの最初から出力されます．最初のクロック(T_1)の立下りから，CPUがデータ・バスにデータを出力し始めるとともに，$\overline{\text{MREQ}}$信号が“L”になります．$\overline{\text{WR}}$はデータよりも1クロック遅れて“L”になり，さらにその1クロック後に“H”に戻ります．データは$\overline{\text{WR}}$が“H”に戻ってから半クロックの間，保持されています．

このように，Z80のメモリ書込みサイクルでは$\overline{\text{WR}}$の前縁に対してもデータ・セットアップ時間が確定されており，出力データ・ラッチとしてDフリップフロップを用いた場合は立上り・立下りのいずれでトリガしてもよいことになります．また，トランスパレント・ラッチを用いることができます．

つぎにI/O書込みサイクルを調べます(図2.12)．メモリ書込みサイクルに比べてバス・

*　メモリなどでは，このほかにアドレス・アクセス時間が重要となる．

図2.11　Z80Aのメモリ書込みサイクル

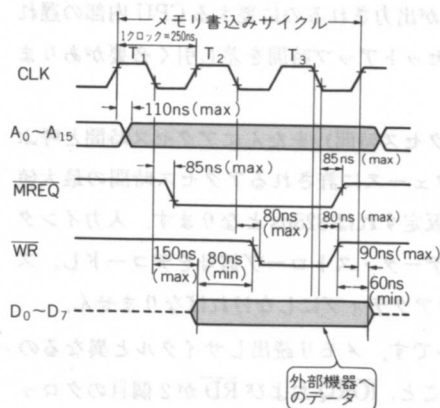
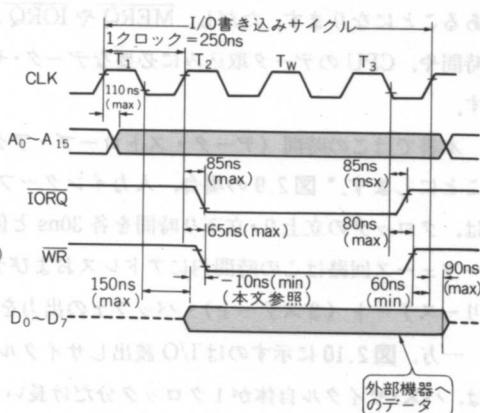



図2.12　Z80AのI/O書込みサイクル

サイクル全体の長さが1クロック分だけ長くなっていることのほか，データが出力されてから$\overline{\mathrm{WR}}$が“L”になるまでの時間が短くなっている点に注意を要します．

このため，$\overline{\mathrm{WR}}$の前縁に対するデータ・セットアップ時間が最悪値で−10ns，すなわちデータが確定するのが$\overline{\mathrm{WR}}$の前縁より遅くなることがあり得るのです.*したがって，Z80のI/O領域に出力ポートを設ける場合，原則として$\overline{\mathrm{WR}}$の後縁でDフリップフロップをトリガすることになります．

⑶　**M1サイクル**　　各命令のOPコード読出しの際にのみ使われるのがこのバス・サイクルです．

図2.13に示すように，$\overline{\mathrm{MREQ}}$および$\overline{\mathrm{RD}}$が通常の読出しサイクルよりも早く“L”になります．CPUがデータを取り込むのは第3クロック目の立上りであり，データ・ストローブ・アクセス時間は215nsときびしいものになります．

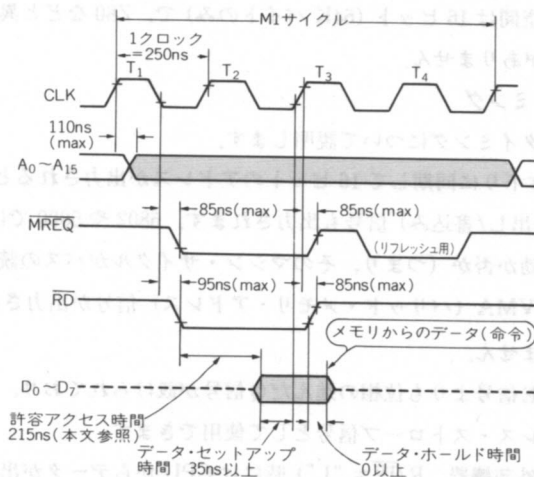
ただし，通常は入出力インターフェースでこのサイクルを使用することはありません．また，割込み処理の際に最初にM1サイクルに入りますが，このときはCPUが自動的に2クロックの延長を行うので，アクセス時間の問題は少なくなります．

以上，Z80A CPUのバスのタイミングの概略を説明しました．

ハードウェアの設計には以上のような内容を必ず知っておかなければなりません．計算

---

*　Z80Aでもクロック周波数を低くするか，クロックのデューティ比を変えることにより，セットアップ時間を確保する方法はある．

図2.13　Z80AのM1サイクル($\overline{M1}$および$\overline{RFSH}$は省略する)

自体はきわめて簡単（タイミング計算には足し算と引き算としか使わない）ですが，これを何回も行うのは面倒です．そこで，よく使うCPUについて何通りかの条件を設定し，重要な数値をあらかじめ計算して表にしておくことをおすすめします．

なお，図2.9～図2.13に示した数値はZ80A CPUをデューティ比50％，4MHzのクロックで用い，クロックの立上りおよび立下り時間を30nsと仮定した場合のものです．バス負荷条件はI_{OL}=1.8mA (V_{OL}=0.4V) および50pFです．バスの容量性負荷がこれより大きくなると，各信号の遅れ時間も増します．このあたりは使用するCPUのデータ・シートをよく読んでかかることです．

2.4　各種8ビットCPUのバス・タイミング

● 68系（6809/6802/6800）のバス

6809，6802，6800などのバスはシステム・クロックであるE信号*に同期して動作します．この信号の1周期がそのまま1マシン・サイクルであり，バス・サイクルでもありま

＊　6800ではϕ2に対応する．ただし，6800のバスのタイミングはϕ1を基準として定められている．また，このE(またはϕ2）は一般にはデータ・ストローブ（DS）信号に相当する．

す．この点，1マシン・サイクルの長さが変化するZ80としばしば対比されます．

また，アドレス空間は16ビット（64Kバイトのみ）で，Z80などと異なりメモリ領域とI/O領域との区別がありません．

▶ 68系バスのタイミング

つぎに，バスのタイミングについて説明します．

まず，E信号の立下りに同期して16ビットのアドレスが出力されるとともに，同じタイミングでR/$\overline{W}$（読出し/書込み）信号も出力されます．6802や6800では，アドレス・バス上のアドレスが有効か否か（つまり，そのマシン・サイクルがバスの読み書きを必要とするか否か）を示すVMA（バリッド・メモリ・アドレス）信号が出力されますが，6809ではこの信号はありません．

一方，6809ではE信号よりも位相の進んだQ信号が設けられており，QとEとのORをとった信号をアドレス・ストローブ信号として使用できます．

書込み（CPU→外部機器，R/$\overline{W}$＝“L”）時にはCPUからデータが出力されます．一般にはE信号とR/$\overline{W}$信号とにより書込みストローブ信号をつくります．ただし，E信号の立下りから出力データが保持されている時間は30ns（min）と短いので，周辺回路の設計のいかんではデータ・ホールド時間が不足します．6809ではQ信号を用いればなんとかなりますが，6800や6802では設計時に慎重な計算を必要とします．

また，読出し（外部機器→CPU，R/$\overline{W}$＝“H”）時には，CPUはE信号の立下りでデータを取り込みます．このタイミングの前後にはデータ・セットアップ時間とデータ・ホー

図 2.14　68B09のバス・タイミング（5 V±5％，0〜70℃）

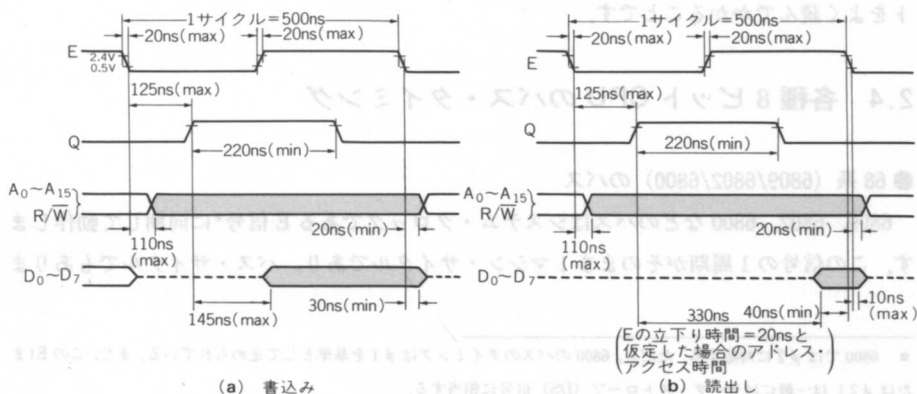
(a) 書込み　(b) 読出し

ルド時間の両方が必要です．実用上，デコーダやバッファの遅延時間によってデータ・ホールド時間が確保されますが，最悪値設計をするとCPUに対するデータ・ホールド時間が不足することがあるので注意します．

図2.14に68B09をクロック2MHzで動作させたときのタイミング(最悪値)を，図2.15（次ページ参照）に基本的な入出力インターフェースの接続方法をそれぞれ示します．

なお，バス負荷条件はI_{OL}＝2mA(V_{OL}＝0.5V)およびC_L＝130pF(ただしアドレスA_0～A_{15}は90pF)です．

● 8048系（8048～8051/8031/35/39など）のバス[4]

8048シリーズ(RAMおよびROM内蔵の8048～8051，そのEPROM版の87××，ROM外付けの803×）はインテル社の代表的なシングルチップ・コンピュータで，CMOSタイプも含めて多くのセカンド・ソースがあり，モステック3870系と並んで非常に広く使用されています．

このCPUは，外部メモリや周辺装置が不要の場合は24本の端子を入出力ポートとして使えますが，外部メモリなどを接続する場合はその一部(たとえば12本)をバスとして使用します．その場合，8本はデータとアドレスとの時分割バスとなっており，結局データ8ビット，アドレス12ビットとして使用することになります．また，プログラム・メモリ領域とデータ・メモリ（およびI/O）領域とが分かれているので注意が必要です．

▶ 8048系バスのタイミング

まず，命令読出し時(図2.16(a))には，バスにアドレスが出力されますので，ALEの立下りでこのアドレスをラッチして使用します．ラッチはトランスパレント・ラッチでもDフリップフロップでもかまいませんが，トランスパレント・ラッチのほうがt_{AL}(11MHz発振時に50ns)だけアクセス時間を長くとれます．つぎに，命令読出しストローブ信号として$\overline{\text{PSEN}}$が“L”になりますから，プログラム・メモリからデータ（プログラム）を出力します．CPUは$\overline{\text{PSEN}}$の立上りでこれを取り込みます．

一方，データ・メモリや外部I/Oポートの読出し時を図2.16(b)に示します．この場合は，読出しストローブ信号として$\overline{\text{RD}}$が“L”になります．先程の$\overline{\text{PSEN}}$とこの$\overline{\text{RD}}$とは出力されるタイミングが少し異なっており，$\overline{\text{RD}}$に対するアクセス時間のほうが余裕があります．

最後にデータ・メモリや外部I/Oポートに対する書込み時を図2.16(c)に示します．書込みストローブ信号$\overline{\text{WR}}$は，$\overline{\text{RD}}$と同じタイミングで出力されます．$\overline{\text{WR}}$の立下り時には

CPUからのデータはまだ出力されていませんので，$\overline{\text{WR}}$の立上りを用いて書込みを行わなければなりません．

なお，図2.16に示したのはインテル社（オリジナル・メーカ）のHMOS版の8048（8048HL-1）を11MHzで使用したときのタイミングです．相当品種もクロック発振周波数が同じならば大体似たようなタイミングですが，NMOSかCMOSかにより，あるいはメーカにより，細かい数値は相当異なりますので，実際に使用するときは注意が必要です．

バスの負荷は$\overline{\text{PSEN}}$，$\overline{\text{RD}}$，$\overline{\text{WR}}$，ALEが$I_{OL}=1.8\text{mA}$（$V_{OL}=0.45\text{V}$）および$C_L=80\text{pF}$，その他のバスは$I_{OL}=2\text{mA}$（$V_{OL}=0.45\text{V}$）および$C_L=150\text{pF}$です．基本的な入出力方法の例（普通，このテのシングルチップ・コンピュータでこのような回路を外付けする必要はないが…）を図2.17に示します．

図2.15　68B09と入出力ポートとの接続

図2.16　8048のバス・タイミング（5 V±10％，0～70℃）

（a）命令の読出し

（b）データ・メモリ，I/Oポートの読出し　　（c）書込み

●Z8系（86××）のバス[3]

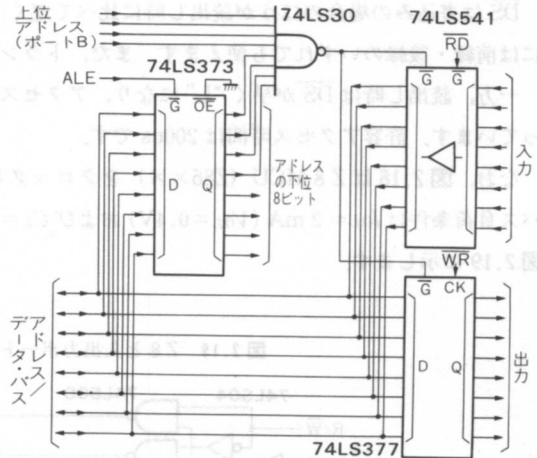
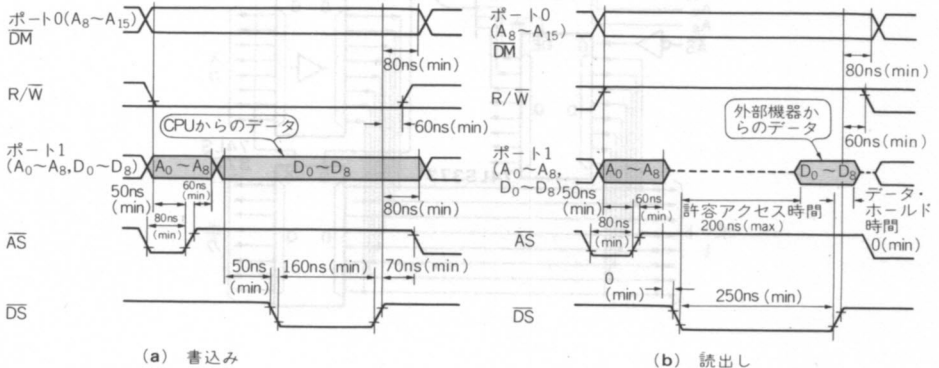
ザイログ社のシングルチップ・コンピュータZ8シリーズは，Z80などの汎用8ビットCPUに比べて機能的に一段上であり，8ビット系のコントローラとして今後が期待されています．

命令は16ビットのZ8000のそれに似ており，バス構成もZ8000とともに“Z-BUS”と呼ばれています．ただし，Z8はもともとシングルチップ・コンピュータですのでI/O端子の一部をバスとして使用する点は8048などと同じです．

アドレス空間はデータ領域と命令領域とに分かれており，各々最大64Kバイト（合計

図2.17　8048と入出力ポートとの接続

図2.18　Z8のバス・タイミング

（a）書込み　（b）読出し

128Kバイト）が使えます．

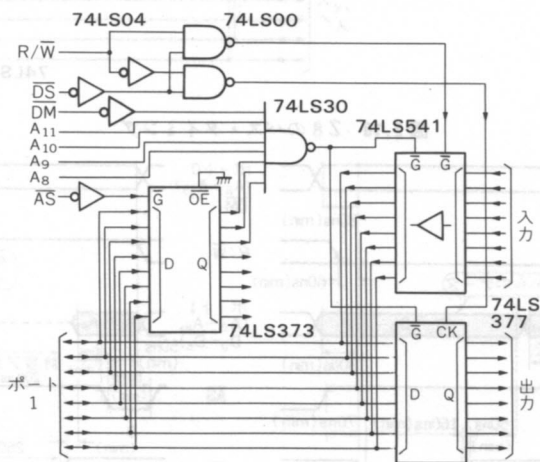
▶ Z8系バスのタイミング

図2.18(a)に書込みタイミング，同図(b)に読出しタイミングをそれぞれ示します．バス・サイクルが始まると，ポート0に上位4または8ビット，ポート1に下位8ビットのアドレスが出力されます．同時に$\overline{AS}$（アドレス・ストローブ）が“L”になりますので，トランスパレント・ラッチを用いて下位8ビットのアドレスをラッチします（Dフリップフロップも使用可能）．R/$\overline{W}$（読出し/書込み）および$\overline{DM}$（データ領域を示す）はアドレスと同時に出力されます．このあと，$\overline{DS}$（データ・ストローブ）が“L”になるので，データの転送を行います．

$\overline{DS}$は書込みの場合のほうが読出し時に比べて遅く“L”になり，書込みデータのラッチには前縁・後縁のいずれでも使えます．また，トランスパレント・ラッチも使用できます．

一方，読出し時は$\overline{DS}$が早く“L”になり，アクセス時間が少しでも長くできるようになっています．許容アクセス時間は200nsです．

なお，図2.18はZ8 MCU（Z86××）をクロック8MHzで使用した場合のものです．バス負荷条件はI_{OL}= 2mA (V_{OL}=0.4V) およびC_L=150pFです．基本的な入出力の例を図2.19に示します．

図2.19 Z8と入出力ポートとの接続

● 8088（ミニマム・モード），8085，NSC 800のバス[(4)(16)]

8085はインテル社の8ビットCPUで8080Aの後継機，8088は8086と命令互換性のある8ビットCPU，NSC800はZ80と命令互換性のあるNS社のCMOS 8ビットCPUです．

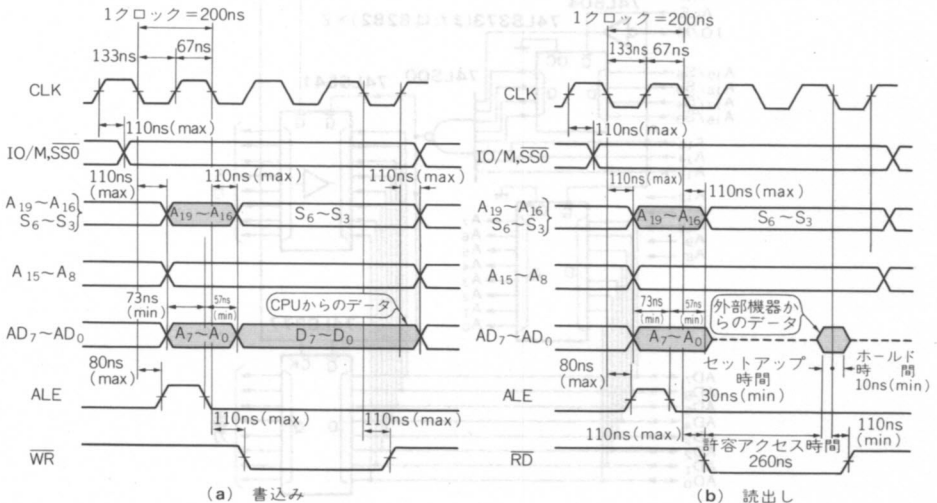
8088にはミニマム・モードとマキシマム・モードとがありますが，そのうちミニマム・モードを用いれば，この3種のCPUは非常によく似たバス構成となっています．ここでは8088を例にとって，そのタイミングを説明します．

8088（ミニマム・モード）は20ビットのアドレスと8ビットのデータをもち，下位8本のアドレスはデータと，上位4本のアドレスはステータス信号とそれぞれ多重化されています．

▶ 8088，8085，NSC 800のバスのタイミング

図2.20(a)に示すのは書込みのタイミングです．8088のクロック信号は最適デューティ比が1/3で，その許容範囲が狭いのが難点です．各時分割バスのアドレス(A_{19}～A_{16}，A_7～A_0）は，ALE（アドレス・ラッチ・イネーブル）信号でラッチします．A_{15}～A_8は時分割ではありませんが，インテル社ではラッチを付けることを推奨しています．書込みストローブ$\overline{WR}$の前縁（立下り）ではデータは保証されませんから，必ず$\overline{WR}$の後縁（立上り）

図2.20　8088（ミニマム・モード）のバス・タイミング（5 V±10％，0～70℃）

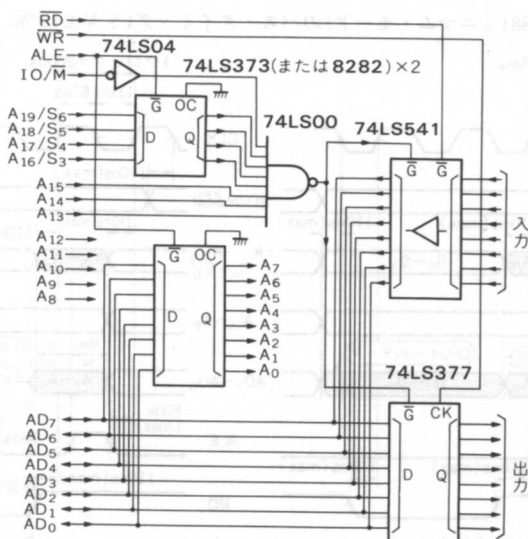
（a）書込み　（b）読出し

でDフリップフロップにデータを書き込みます．

一方，図2.20(b)に示すのは読出し時のタイミングです．$\overline{RD}$信号は$\overline{WR}$信号と同じタイミングで出力されます．$\overline{RD}$からの許容アクセス時間は約260ns（クロック5MHz）です．

なお，図2.20に示す数値はクロック周波数5MHz，デューティ比1/3，立上り，立下り時間は無視したものです．バス負荷はI_{OL}＝2mA（V_{OL}＝0.45V）およびC_L＝100pFです．基本的な入出力ポートの接続を図2.21に示します．

以上，いくつかの8ビットCPUのバス・タイミングを説明しました．これ以外にも多くのCPUがあるのですが，ハードウェアの詳しい資料が入手できれば，入出力インターフェースの設計は容易でしょう．

一方，今後は16ビットや32ビットのCPUが制御用にも使われるようになるでしょうが，基本的には何ビット機であってもハードウェアにかぎってみれば同じことです．ただし，16ビット以上のCPUでは，1語（たとえば16ビット）のうちいずれかの8ビットだけを扱う場合と，1語全部を扱う場合とがあり，そのための制御信号の出方に各CPUの特徴が見られます．

図2.21　8088と入出力ポートとの接続

また，CPUが高速になるにしたがい，メモリや入出力インターフェースがアクノリッジ信号を返さないかぎり，CPUが待ち続ける回路構成を採用することも増しており(Normally-Waitなどとも呼ぶ. 68000では基本的にこの形)，インターフェースの設計にも多少の考慮が必要です．

しかし，いずれの場合もCPUの動作をよく理解すれば，設計はきわめて容易だという点は覚えておいてください．

コラムB　バスの種類

●データ/アドレス/コントロール

普通，CPUのバスはデータ・バス，アドレス・バス，コントロール・バスなどと分類します．いうまでもなく，読み書きするデータを乗せるのがデータ・バス，アドレスを指定するのに使うのがアドレス・バス，読出しや書込みのストローブ信号や割込み線などをコントロール・バスと呼びます．

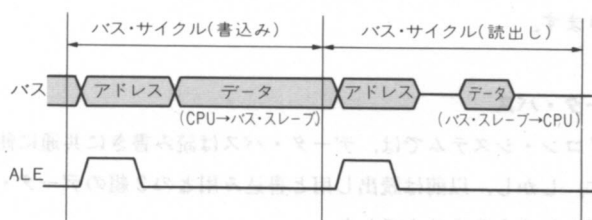
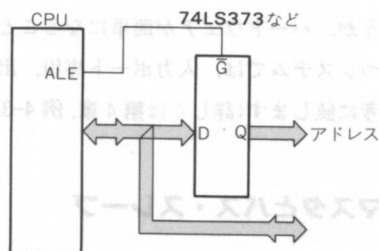
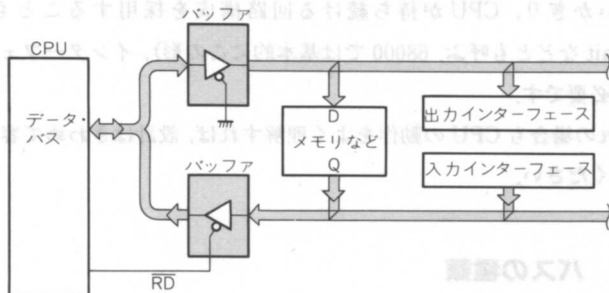
図B.1　時分割バスとその分離

(a)　タイミング

(b)　ALE信号の使用方法

図 B.2 一方向のバスの1例

●時分割（多重化）バス Multiplexed bus

CPU はすでにご承知のように，バスの本数が非常に多く，さらに高機能の CPU ほどその本数が増える傾向にあります．そこで，配線の本数を減らすため，バスの時分割（多重化）が行われます．

最も多いのはデータとアドレス（の一部）との多重化です．これは図 B.1 のように，バス・サイクルの初めにはアドレスを乗せ，そのあとはデータ・バスとして使うものです．図 B.1 のように，ALE 信号を用いてラッチすれば，アドレス信号を分離することができます．

●双方向データ・バス

現在のマイコン・システムでは，データ・バスは読み書きに共通に使う（双方向）のが普通です．しかし，以前は読出し用と書込み用との2組のデータ・バスを持つシステムはめずらしくありませんでした．

今でも，入力と出力とが別々の端子になっているメモリを使う場合など，部分的に専用のバスを使ったほうが，ハードウェアが簡単になることがあります（図 B.2）．また，多数の入出力を持つシステムでは，入力ポート専用，出力ポート専用といったバス・バッファ方法は一考に値します（詳しくは第4章，例 4-3 などを参照して下さい）．

コラムC バス・マスタとバス・スレーブ

バスには様々な要素が接続されますが，そのうちバスを管理するものをバス・マスタと呼びます．通常，アドレス信号や制御信号はバス・マスタが出力します．一方，

バス・マスタ以外のものをバス・スレーブと呼びます．

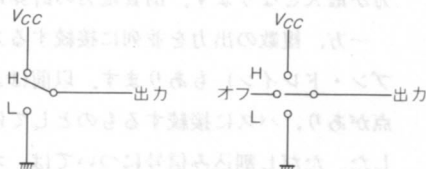
簡単なマイコン・システムでは，CPUがバス・マスタであり，メモリや入出力インターフェースなどがすべてバス・スレーブです．

しかし，DMAコントローラやコプロセッサを含むシステム，あるいは複数CPUシステムでは，バス・マスタとバス・スレーブとの関係は複雑になります．

コラムD　スリーステート

計算機のデータ・バスのように同じ信号線に何本もの出力線が接続される場合，2本以上が同時にバスに信号を出力すると衝突が起こるので，不要な出力をオフ状態にしなければなりません．通常の論理値（H，L）のほかに第3の状態（オフ）を持つので，この機能をスリーステート（three-state）と呼びます（トライステートは商標）．

スリーステート機能は，図D.1のようなものだと考えればわかりやすいでしょう．TTLでは74LS125，126が有名です（図D.2(a)）．また，4ビットまたは8ビット単

図D.1　スリーステート機能の概念

（a）通常のディジタル出力（b）スリーステートの出力

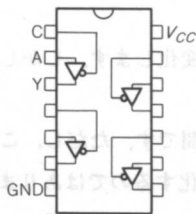
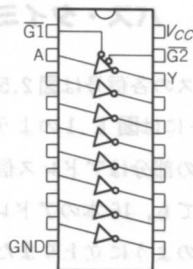
図D.2　スリーステート出力のICの1例[5]

（図は74LS125．C＝"H"で出力オフとなる．74LS126はC＝"L"で出力オフ）
（a）74LS125/126

（図は74LS540．$\overline{G1}$＝Hまたは$\overline{G2}$＝Hで出力オフ．74LS541は非反転）
（b）74LS540/541

表 D.1　駆動能力の比較[5]

		通常のゲート（74LS00）	スリーステートのゲート（74LS125A）	専用のバス・ドライバ（74LS540）	
$I_{OH(max)}$	V_{OH}≧2.7V	−0.4			mA
	V_{OH}≧2.4V		−2.6	−3	
	V_{OH}≧2.0V			−15	
$I_{OL(max)}$	V_{OL}≦0.4V	4	12	12	mA
	V_{OL}≦0.5V	8	24	24	

位のスリーステート・バッファもあります．その1例として74LS540，541の端子接続を図D.2(b)に示します．

一般に，スリーステート出力はバスに接続されることが多いので，駆動能力も大き目になっています．表D.1にその1例を示します．なお，TTLの場合，スリーステート機能を持つものは消費電力がかなり大きく，しかも出力がオフ状態のときに消費電力が最大となります．消費電力の計算にはくれぐれもご注意下さい．

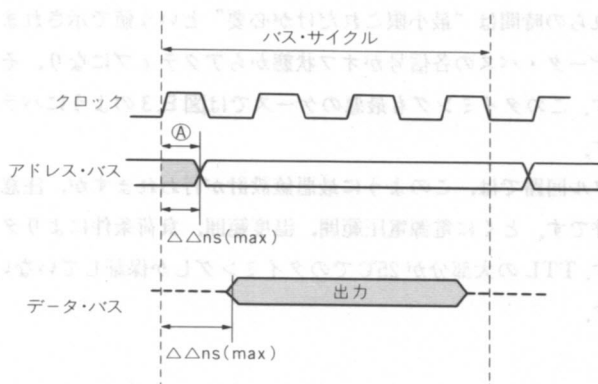
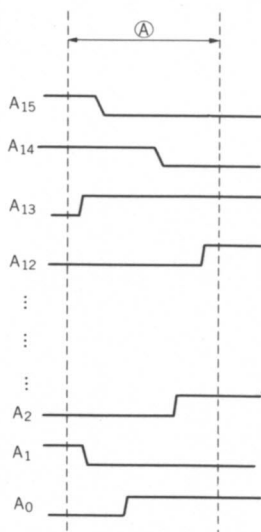
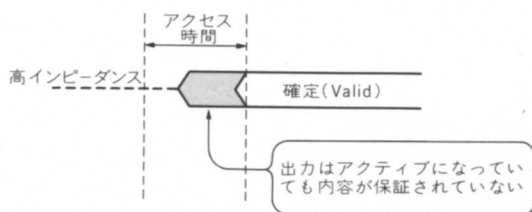
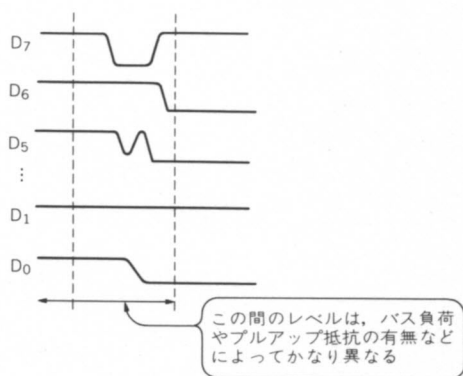
一方，複数の出力を並列に接続する方法として，オープン・コレクタ（またはオープン・ドレイン）もあります．以前はよく使われたものですが，速度が遅いなどの欠点があり，バスに接続するものとしては，スリーステートに置き換えられてしまいました．ただし割込み信号については，オープン・コレクタ（またはオープン・ドレイン）のワイヤードOR機能を生かして利用されることが少なくありません．

コラムE　バス・タイミングの記述方法

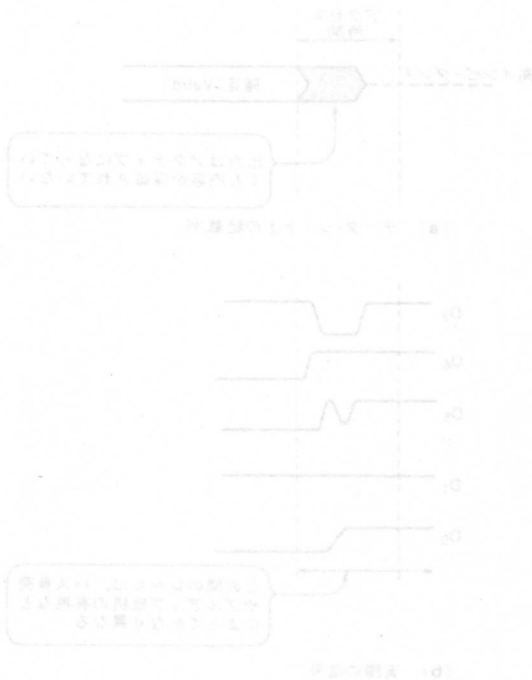
実際のバスの各信号は図2.5（24ページ参照）のように変化します．しかし，ICのデータ・シートには図E.1のような書き方がされています．

図中，Ⓐの部分はアドレス信号（16本）の最大遅延時間です．ただし，このように書いてあっても，16本のアドレス信号がすべて同時に変化するのではありません．実は，図E.2のように立上りまたは立下りのタイミングが0～△△nsの間にバラついているのです．

一方，データ・バスは出力と入力（スリーステート時を含む）との両方の状態があ

図E.1　データ・シートに記載されているタイム・チャートの例

図E.2　図E.1Ⓐ部の実際

図E.3　スリーステート出力タイミング

(a)　データ・シート上の記載例

(b)　実際の信号

ります．入力の場合，データ・セットアップ時間およびデータ・ホールド時間が必要です．これらの時間は“最小限これだけが必要”という値で示されます．一方，出力の場合，データ・バスの各信号がオフ状態からアクティブになり，そのあとデータが確定します．このタイミングも最悪のケースでは図E.3のようにバラついていると考えられます．

ディジタル回路では，このように最悪値設計が行われますが，注意したいのはその特性の条件です．とくに電源電圧範囲，温度範囲，負荷条件によりタイミング特性は異なります．TTLの大部分が25℃でのタイミングしか保証していないのは，本当は困りものです．

第3章

マイコン入力インターフェースの実際

ここでは外部機器からマイコンにデータを入力する方法を示します．

まず，この入力用インターフェースの一般形を図3.1に示します．外部機器からのディジタル信号は，①レベル・シフタによって本来のレベルおよび形態（たとえば接点信号）から所定のレベル（たとえばTTLレベル）に変換し，②必要に応じてラッチやデコーダを通し，③バス・インターフェース部分を経て，④マイコンのバスに接続します．アナログ信号の場合，適切なレベルに変換してからアナログ-ディジタル変換器（A-Dコンバータ）によりディジタル信号にかえ，あとはディジタル信号として処理されます．

この章では以上の各部について事例を紹介しながら設計法を説明します．

3.1　入力信号の種類と処理法

●入力信号の形態とレベル

マイコンに接続する入力信号は，アナログ信号とディジタル信号とに大別できます．マイコンがアナログ信号を取り込むには，コンパレータなりA-D変換器なりを通してディジタル信号に直してから，ディジタル信号として入力することになります．

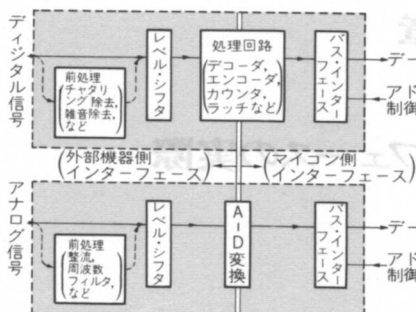
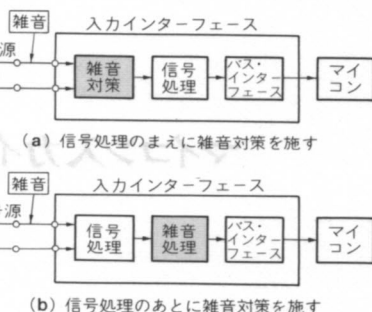
ディジタル信号として使用する機会が多いのはTTLレベルや接点，あるいはオープン・コレクタなどの信号です．ただし，細かく見れば，たとえTTLレベルであっても許容ファン・アウトの違いから，接続可能なものとそうでないものとが生じます．また，接点信号はチャタリング特性の違いや，接点の種類（微小信号用か否か）などにより回路上の工夫が必要です．

RS-232C，RS-422や，IEEE-488（GPIB）のように規格が定まっているものに対して

図3.1　入力用インターフェースの一般形

図3.2　雑音対策の挿入位置

(a) 信号処理のまえに雑音対策を施す

(b) 信号処理のあとに雑音対策を施す

は，専用のレベル変換ICが使えます．それ以外のものとしてCMOS，PMOS，HTL，リレー論理回路など，レベルの違うディジタル信号があります．これらは個々にレベル変換回路を設ける必要があります．ただし，ハードウェアの標準化という観点から，なるべく広い範囲のレベルに対して使用できる回路を考えておくのは有効です．

以上の問題とは別に，先に触れた雑音の問題により，アイソレーション（絶縁）や雑音処理を行わなければならない場合があります．一般に，このような処理はなるべく信号源に近い部分で行ったほうが望ましいといえます（図3.2参照）．

一方，アナログ記号は各種のセンサやトランスジューサ，あるいは測定機器からの信号と考えられます．簡単なものではコンパレータで“H”か“L”かに分けてしまえばよいのですが，一般的にはA-D変換することになります．

A-D変換器自体は非常に高性能のものが入手できますが，マイコンという大きなディジタル回路と共存する点が問題となります．たとえば，12ビットのA-D変換を行うと，アナログ信号の分解能は10～100μVという微小なものになり，わずかなディジタル雑音の混入も大問題となります．

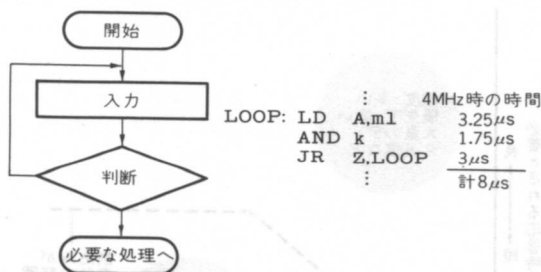
以上のように，入力信号の形態およびレベルの違いが，入力インターフェースの外部機器側の設計に大きく影響します．

●入力信号の速度

マイコンが信号を入力する速度には，少なくともつぎの2種類があります．

(1)　信号が印加されてからそれを検出するまでの応答時間．

(2)　信号を処理し終わるまでの処理時間，あるいは連続的に処理を繰り返せる処理周期．

図3.3 信号の判断(ポーリング・ループ)

マイコンに接続する信号のなかには，何年かに一度しかアクティブにならない（あるいは永久に使用されることのない）異常検出信号のようなものがあります．ただし，異常発生後1秒以内に対策をとらねばならないとすれば，少なくとも応答時間は1秒以下でなければなりません．しかし，処理周期はこれほど短い必要はありません．

一方，FIFO（First-In First-Out：先入れ先出しメモリ）を介して入力されるデータ列を読み込むような場合，平均的な処理周期がデータ転送速度よりも速いことが重要です．

よく知られているように，信号の有無を知るには割込みとポーリング*との2種類があります．先程の区分から見ると，割込みを用いれば応答時間を早くすることができます．ただし，処理時間については大差ありません．ちなみに，ソフトウェアで達成できる応答速度はどの程度でしょうか．簡単なループを図3.3に示しますが，4 MHzのZ80Aでこのループを1回まわるのに8 μsが必要です．ループ内に処理や判断が追加されれば，これより遅くなります．

一方，割込みの場合はどうでしょう．一般に8～16ビットの汎用CPUで多く使われている割込み方法では，CPUが割込みを受け付けてから実際の処理ルーチンに入るまでにつぎのような手順が必要です．

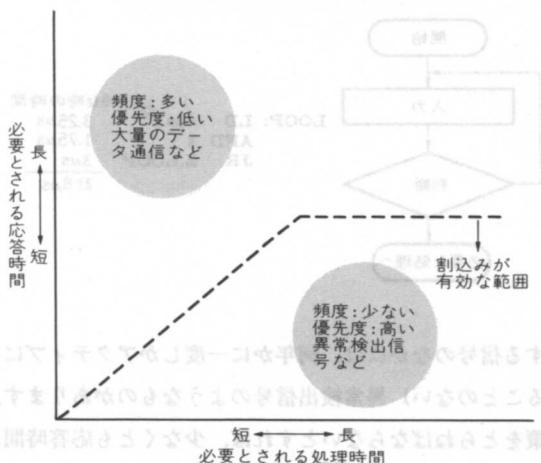
(1) 実行中の命令のアドレスをスタックに退避する．

(2) 外部機器または割込みコントローラから割込みベクタ（あるいは機器番号）を入力する．

(3)割込み用のエントリ・ポインタ・テーブルを参照して処理ルーチンを呼ぶ．

---

* ここでいうポーリングとは，ソフトウェア・ループで信号を判断することをいう．本来は複数の信号を順々に調べることをポーリングという．

図 3.4　応答時間と処理時間との関係

このほか，必要なレジスタの退避を含めると，一回の割込みについて数10～100μsくらいのオーバヘッドが必要です．つまり，応答時間は100μsのオーダとなり，ポーリング・ループと大差ありません．ただし，CPUが他の仕事をしていてポーリングができない場合でも割込みは可能ですから，やはり応答時間を重視する場合は有効です（図3.4を参照）．

このほか，多くのCPUにはDMA（ダイレクト・メモリ・アクセス）機能があります．ただし，これは機能とはいっても，CPUが仕事を休んで仕事場を他人に貸しているのですから，外部ハードウェアの問題となります．

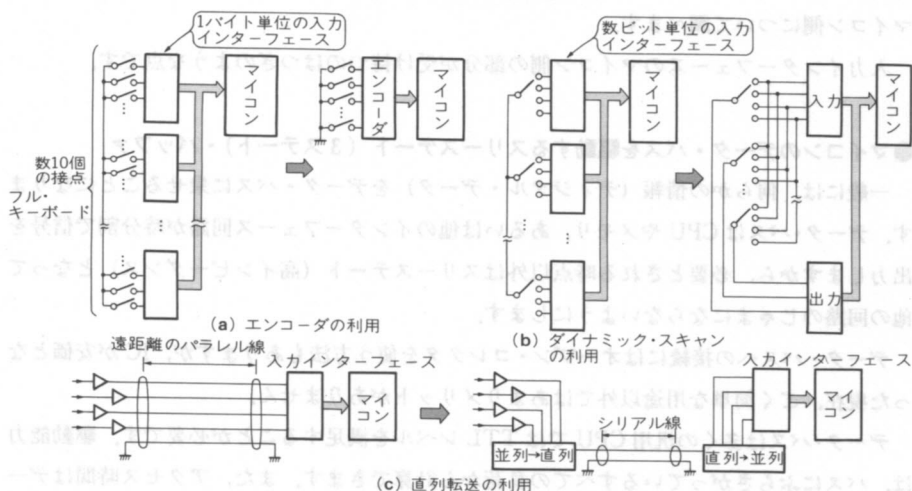
以上に説明した速度よりも速い処理が必要な場合，インターフェースの外部機器側ハードウェアの出番となります．また，遅い信号であっても様々な理由からハードウェアとしたほうがよい場合もあります．これらについては本章3.3で詳しく説明します．

●入力信号の数と処理方法

入力信号の数が多くなった（数10本以上）場合，少数の信号（たとえば8本＝1バイト）の入力用インターフェースをたくさん作って並列に接続しただけではいけないのでしょうか？

もちろん，ちゃんと設計されたものならば何回路接続しても動作するはずです．しかし，その場合につぎの二つの問題が生じます．

(1)　**ハードウェア上の問題**……多くの回路を接続することにより，バス負荷や消費電力

図 3.5 多数の信号を取り込む工夫

が増加し，設計が困難となる．

⑵ **ソフトウェア上の問題**……前項で述べた入力速度は，入力信号数を増やせばさらに低下する．また，個々の処理も複雑となる．

これらの問題に対して，つぎのような対策がとれないかを考慮してみます．

⑴ エンコード（符号化）あるいは簡単な論理演算で信号数が減らせないか？

⑵ ダイナミック・スキャンにより信号数を減らせないか？

⑶ 優先順位がつけられないか？

⑷ 直列（ビット・シリアル）に変換できないか？

このうち⑴の方法によりバス・インターフェースを簡単にできます（図 3.5(a)）．また，⑵の方法は外部機器側の配線を少なくします（図 3.5(b)）．たとえば50個以上の接点をもつフル・キーボードでは⑴，⑵を併用しているのが普通です．また，多数のロータリ・スイッチの読込みには⑵の方法が，遠距離の信号源を見るには⑷の方法がそれぞれ有効です（図 3.5(c)）．

これらの事例については，このあと 3.2 と 3.3 とで詳しく説明します．

3.2 入力インターフェースのマイコン側の設計

入力インターフェース回路が必ずしも二分できるわけではありませんが，図 3.1 に書い

たように，マイコン側と外部機器側とに分けて考えてみるのが便利です．ここでは，まずマイコン側について調べます．

入力インターフェースのマイコン側の部分が受け持つのはつぎのような点です．

●マイコンのデータ・バスを駆動するスリーステート（3ステート）・バッファ

一般には，何らかの情報（ディジタル・データ）をデータ・バスに乗せることになります．データ・バスはCPUやメモリ，あるいは他のインターフェース回路が時分割で信号を出力しますから，必要とされる時点以外はスリーステート（高インピーダンス）となって他の回路のじゃまにならないようにします．

データ・バスへの接続にはオープン・コレクタを使う方法もありますが，ICが安価となった現在，ごく簡単な用途以外ではあまりメリットがありません．

データ・バスは多くの汎用CPUではTTLレベルを満足することが必要です．駆動能力は，バスにぶらさがっているすべての負荷から計算できます．また，アクセス時間はデータ・ストローブ信号（$\overline{RD}$など）に対して100～500nsのものが多く，TTLのスリーステート・バッファを使えば，たいていは要求を満足できます．

ただし，特殊なレベルや速度を要求するCPUもありますから，注意が必要です．

●アドレスや制御信号が所定の組合せとなったことを検出するデコーダ

上記のスリーステート・バッファの出力をアクティブにするのは，アドレスや制御信号が所定の組合せになった時点だけです．これを検出するためデコード回路が必要となります．

CPUのアドレス領域を有効に使うにはフル・デコード（コラムG）とすべきですが，実際には必要に応じて単純化します．また，デコーダの設計によって取込み方法（バイト単位かビット単位か，など）が決定されるわけで，ソフトウェアとの兼ね合いも考えます．

デコーダはCPUのアドレス・バスや制御バスの負荷になるとともに，アクセス時間に影響する部分ですから，タイミング設計時には注意が必要です．

例3-1　基本的な入力インターフェース

マイコンへの入力は，データ・バスに信号を乗せることが基本となります．これにはアドレスや制御信号が所定の値となったことを検出するデコーダと，必要なときだ

けデータ・バスに信号を乗せるスリーステート・バッファとを用います.

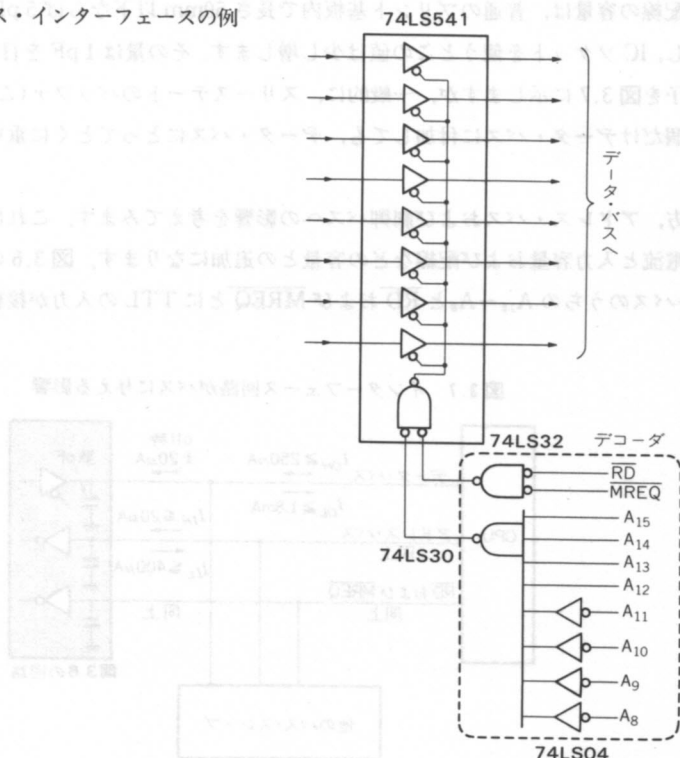
様々な外部機器からの信号（アナログ・ディジタルを問わない）は適切に処理したうえで上記の回路に接続すれば，基本的にはデータの読込みが可能となるわけです.

本例ではこの基本的なバス・インターフェースの設計方法を説明します.

▶ 8ビットの信号を1バイトとして並列に取り込む回路　入力用のバス・インターフェース部分の例を図3.6に示します.

バス・インターフェースとしては，バスに接続できるようなスリーステートの出力をもち，所定のアドレスおよび制御信号が印加されたときのみ出力をアクティブにするデコーダ（コラムF，G参照）を付加する必要があります.

同図は8ビットの入力信号(TTLレベル)を1バイトとして取り込むものです．CPUは

図3.6 入力用バス・インターフェースの例

Z80を仮定しています．デコーダは特定の信号の組合せを検出すればよいのですから，様々な方法が考えられます（コラムF）．

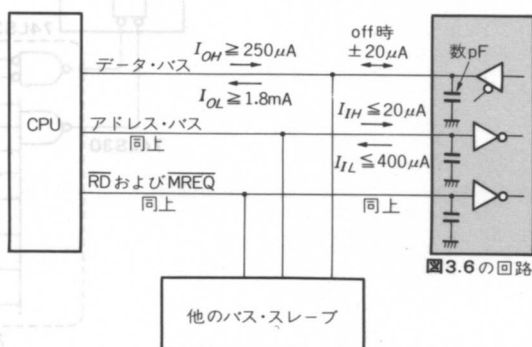
▶マイコンのバスに接続するためには　つぎに，この回路がマイコンのバスに実際に接続できるかどうか調べてみましょう．これは，つぎの2項目に分けて考えてみます．

(1)　この回路を付加することにより，CPUや他のバス・スレーブなどマイコン内部に与える影響．

(2)　この回路がバスを駆動できるか否か．

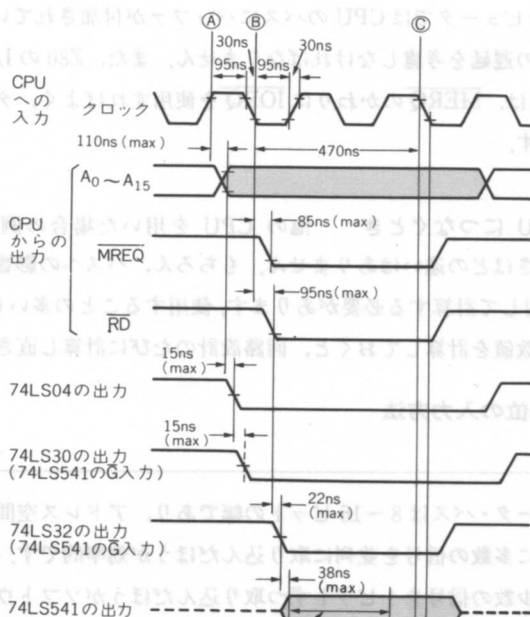
まず，前者はバス負荷の増加分として表されます．図3.6の回路を例にとると，データ・バスについては74LS541の出力漏れ電流（I_{OZL}, I_{OZH}）と出力容量および配線などの容量とが付加されます．74LS541のI_{OZL}, I_{OZH}はそれぞれ20μA（max）です．出力容量はTTLの場合，規格表には明示されていないことが多いのですが，2pF程度と考えて良いでしょう．配線の容量は，普通のプリント基板内で長さ50mm以下ならば5pF以下と考えます．ただし，ICソケットを使うとこの値は少し増します．その量は1pFを目安とします．以上の様子を図3.7に示しますが，一般的に，スリーステートのバッファ（ここでは74LS541）を1個だけデータ・バスに付加しても，データ・バスにとってとくに重い負荷にはなりません．

一方，アドレス・バスおよび制御バスへの影響を考えてみます．これはデコーダ回路の入力電流と入力容量および配線などの容量との追加になります．図3.6の回路では，アドレス・バスのうちのA_{15}～A_8と$\overline{RD}$および$\overline{MREQ}$とにTTLの入力が接続されます．接続

図3.7　インターフェース回路がバスに与える影響

されるのはすべてTTLのゲートであり，$I_{IL}=-400\mu\mathrm{A}$ (max)，$I_{IH}=20\mu\mathrm{A}$ (max) です．入力容量および配線の容量はそれぞれ2pFおよび5pFを目安とします．Z80 CPUの直流的バス駆動能力は“L”レベルで1.8mA，“H”レベルで250μAですから，直流的にはLS-TTLを4個まで駆動できます．したがって，バスに接続されている他の回路との合計が規格内ならば，CPUから図3.6の回路の駆動は可能です．

つぎに，先程の(2)の問題を考えてみます．これには静的なバス駆動能力とタイミングの問題とがあります．74LS541の駆動能力は“L”レベル(0.4V)では12mA，“H”レベル(2.4V)では3mAですから，CPUが直接に(バッファなしに)駆動可能なバスならば，充分に駆動可能です．

▶タイミングの検討　では，タイミングの問題はどうでしょう．図3.6の回路の場合，アドレスと$\overline{\mathrm{RD}}$および$\overline{\mathrm{MREQ}}$とがすべてアクティブとなってから，所定の時間以内にデ

図3.8 インターフェース回路のタイミング計算

ータ・バスにデータを出力する必要があります．図3.8に示すのはこの計算例です．CPUはZ80A，クロックは4MHzでその波形は図に示すように立上り，立下りともに30nsの対称波形であると仮定します．Z80のアドレス・バスはバス・サイクルの最初（図3.8のⒶ）から出力されます．ただし，クロックを基準にすると，Z80の内部回路で最大110nsの遅れが出ます．この信号が図3.6のデコード回路に印加され，一部は74LS04を介して，一部は直接に，それぞれ74LS30を経て74LS541の$\overline{G}$入力に加わります．結局，Ⓐから74LS541の$\overline{G}$までの最大遅延は140nsです．一方，$\overline{RD}$および$\overline{MREQ}$はⒷでアクティブとなり，同様に計算するとⒷから74LS541の$\overline{G}$までの最大遅延は117nsです．74LS541は2本の$\overline{G}$入力がともに“L”となってから出力がアクティブとなりますから，結局，Ⓑからデータ・バスにデータが出力されるまで155nsが必要です．

一方，Z80は©でデータを読み取ります．この際に50nsのセットアップ時間が必要です．つまり，Ⓑから435ns以内にデータが出力されればよいわけで，先程の値はこれを充分に満足していることがわかります．

なお，以上の計算は図3.6の回路をCPUのバスに直接に接続した場合のものです．市販のパーソナル・コンピュータではCPUのバスにバッファが付加されていることが多く，その場合はバッファの遅延を考慮しなければなりません．また，Z80のI/O領域に接続する（I/Oマップト）には，$\overline{MERQ}$のかわりに$\overline{IORQ}$を使用すればよく，タイミング的にはいくぶん楽になります．

▶Z80以外のCPUにつなぐとき　他のCPUを用いた場合の例は第2章，2.4項に示しましたが，さほどの違いはありません．もちろん，バスへの影響やタイミングについては各CPUに対して計算する必要があります．使用することの多いCPUについて主要なタイミング上の数値を計算しておくと，回路設計のたびに計算し直さなくてすみます．

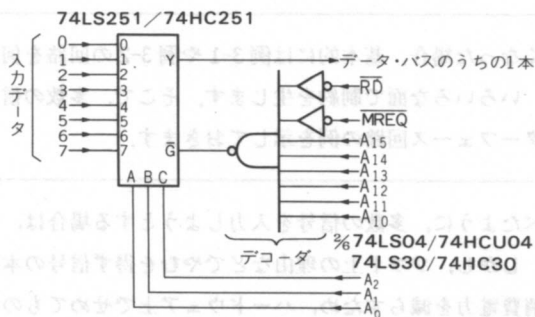
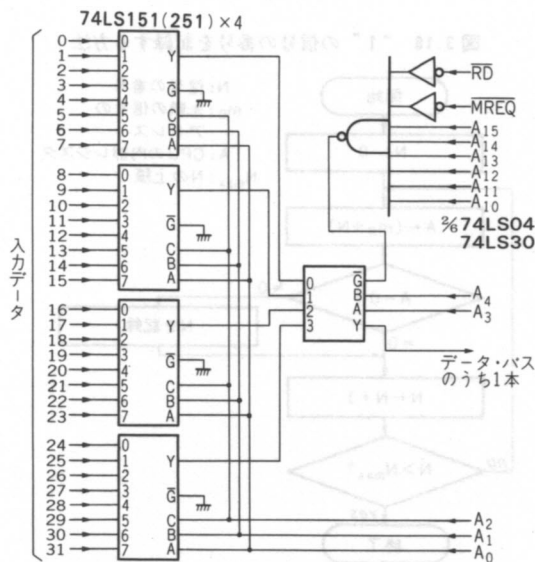
例3-2　ビット単位の入力方法

マイコンのデータ・バスは8～16ビットの幅であり，アドレス空間を有効に使うには例3-1のように多数の信号を並列に取り込んだほうが効率的です．

しかし逆に，少数の信号を1ビットずつ取り込んだほうがソフトウェアが楽になる場合もあります．本例ではそのような場合を説明します．

前項で述べた方法は8ビットの信号を1語として並列に取り込むものでした. 信号を取り込むこと自体は一度に並列に行うのが簡単なのですが, そのうち特定のビットごとに“1”か“0”かを調べなければならない場合, 1つのアドレスで1ビットを取り込んだほうが楽な場合があります.

図3.9に示すのはそのような目的に適した入力用バス・インターフェースの例です. 同図(a)は信号が8本以下の場合, 同図(b)は8本を越える場合を示します. このような方法で

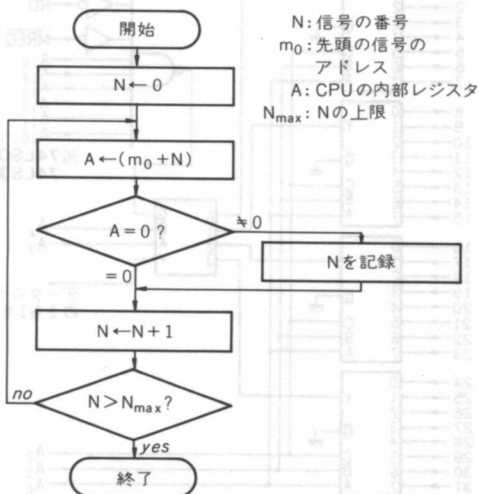
図3.9 ビット単位の入力用バス・インターフェース

(a) 8ビット以下の例

(b) 9ビット以上の例(32ビット)

は，データ・バスの幅を1/8しか利用していないわけで，アドレス空間をムダ使いしていることになります．しかし，たとえば「レベルが“1”のビットの番号を記録する」というような使い方に対しては，このほうが便利です（図3.10を参照）．また，データ・バスへの負荷が軽くなる点や，消費電力が少なくてすむ（TTLの場合，出力がスリーステートになるものは許容ファン・アウトが大きいかわりに消費電力も大きい）利点も見逃せません．

例3-3　多数の信号を入力するインターフェース

信号の数が多くなった場合，基本的には例3-1や例3-2の回路を何回路も接続すればよいのですが，いろいろな面で制約を生じます．そこで，多数の信号を接続するためのバス・インターフェース回路の例を示しておきます．

本章の3.1で述べたように，多数の信号を入力しようとする場合は，信号の本数を減らす工夫が必要です．しかし，ソフト上の理由などでやむを得ず信号の本数が増す場合，バスに対する負荷や消費電力を減らすため，ハードウェア上でせめてもの努力をします．

図3.10　“1”の信号の番号を記録する方法

例3-1は1バイト＝8本の信号を受けることができます．これを4回使えば32本の信号を入力できるようになります（図3.11(a)）．これに対して，マルチプレクサを用いた図3.11(b)の回路も，やはり32本の信号を入力することができます．

▶バッファを用いるかマルチプレクサを用いるか　では，この両者を比較してみましょう．まず，CPUのバスに対する負荷では，

(a) I_{OZL}，$I_{OZH}=\pm20\mu\text{A}\times4=\pm80\mu\text{A}$ (max)

(b) I_{OZL}，$I_{OZH}=\pm20\mu\text{A}$ (max)

消費電力（デコーダ部分を除く）は，

図3.11　32本(4バイト)の入力用バス・インターフェース

(a) バッファを使用した例　(b) マルチプレクサを使用した例

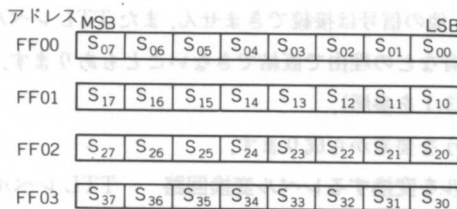
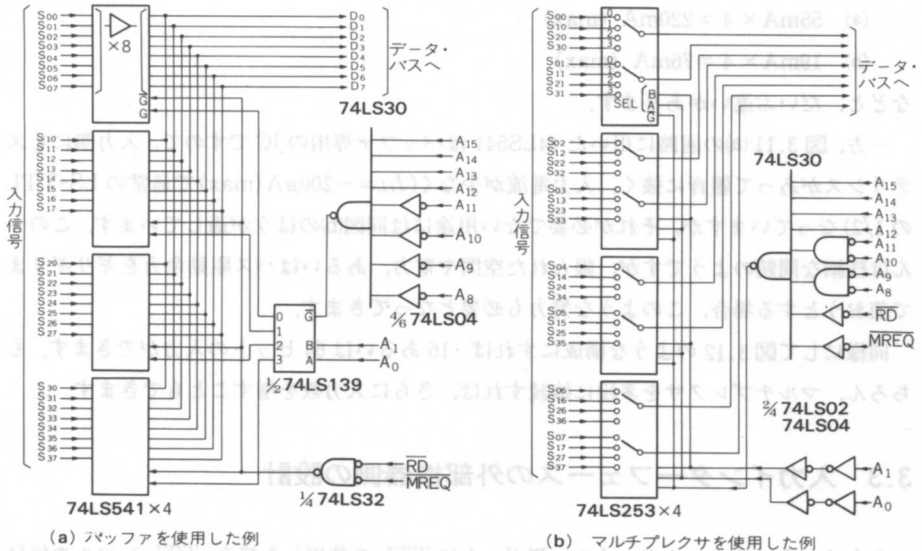
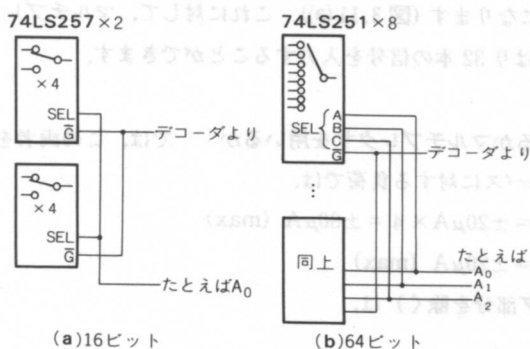
アドレス	MSB							LSB
FF00	S_{07}	S_{06}	S_{05}	S_{04}	S_{03}	S_{02}	S_{01}	S_{00}
FF01	S_{17}	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}
FF02	S_{27}	S_{26}	S_{25}	S_{24}	S_{23}	S_{22}	S_{21}	S_{20}
FF03	S_{37}	S_{36}	S_{35}	S_{34}	S_{33}	S_{32}	S_{31}	S_{30}

(c) 回路(a),(b)のアドレス・マップ

図3.12 マルチプレクサを用いた入力用インターフェース

(a)16ビット (b)64ビット

(a) 55mA×4＝220mA (max)

(b) 19mA×4＝76mA (max)

などと，だいぶ違いがあります．

一方，図3.11(a)の回路に用いた74LS541はバッファ専用のICですので，入力側にヒステリシスがあって雑音に強く，入力電流が少なく($I_{IL}=-200\mu A$(max)で通常のLS-TTLの1/2)なっていますが，それが必要でない用途には同図(b)のほうが適しています．このへんは些細な問題のようですが，限られた空間や電力，あるいはバス駆動余力をギリギリまで使おうとする場合，このような努力も必要となってきます．

同様にして図3.12のような構成にすれば，16あるいは64ビットの入力ができます．もちろん，マルチプレクサを多段に接続すれば，さらに入力数を増すこともできます．

3.3 入力インターフェースの外部機器側の設計

入力インターフェースのマイコン側ポートにTTLを使用した場合，TTLレベルの信号ならば直結できますが，他の信号は接続できません．またTTLレベルであっても変化速度やファン・アウト，雑音などの理由で直結できないこともあります．これらを処理するのが外部機器側です（図3.1を参照）．

この部分はつぎのような要素から成ります．

(1) **信号の形態やレベルを変換するレベル変換回路** TTLレベル以外のディジタル信号であれば，TTLレベル（あるいはマイコン側ポートが必要とする論理レベル）に変換す

る．また，アナログ信号であればコンパレータ（比較器）やA-D変換器を用いてディジタル信号に変換する．

⑵　**信号の変化を吸収するタイミング回路**　変化が速すぎてCPUが扱うのに不適当であれば，ラッチ，ワンショット，カウンタなどを用いて適当なタイミングとなるようにする．

⑶　**信号の本数を減らすエンコード（符号化）回路**　多くの信号を取り扱う場合，簡単な演算で信号の本数が減らせるのであれば，そうしたほうがよい場合がある．また，スキャニングや優先順位についても考慮する．

入力信号の種類は非常に多いため，そのすべてを取りあげることはできませんが，以上の項目を考慮し，代表的なものを以下に示すことにします．

例 3-4　基本的なスイッチの入力方法――チャタリング対策①

マイコンに接続しようとするものの中で，押しボタン・スイッチ，リレー，リミット・スイッチなど，スイッチ類の比重はかなり大きいと思われます．

ところが，スイッチにはチャタリングや接点不良など，特有の問題があります．

本項では，そのような問題をふまえながら，スイッチをバス・インターフェースに接続する方法を説明します．

ここでいうスイッチとは，機械的な接点を示します．したがってリレーは含みますが，オープン・コレクタなどは含みません．

さて，そのような機械的接点の状態を入力する際に大きな問題がいくつかあります．

⑴　**チャタリング**……水銀接点などの例外を除くと，機械的接点には一度にオンまたはオフするのではなく，開閉時に不安定な状態（開閉を繰り返したり，不安定な抵抗値となる）を通過するが，入力インターフェースでは，これを解決する必要がある．

⑵　**スイッチに印加する電圧・電流**……マイコンに入力するには，スイッチにある程度の電圧・電流を印加して，スイッチの開閉を電圧レベルに変換しなければならないが，その際の電圧・電流の大きさを適切な範囲にする必要がある．

以上の2点は本来ならば機構部品屋さんの守備範囲なのですが，実用上きわめて重要ですので少し詳しく説明します．

▶チャタリングの時間はどの程度に見積ればよいか　これは非常によく質問されますが，答えるほうも一口ではいえないほど範囲の広いものです．しかし，ごく大雑把にいえば，小型のマイクロ・スイッチや小型のトグル・スイッチでは5ms±1桁くらいの範囲に入ります．これは，スイッチの製造元にたずねても，あまりはっきりした答は返ってきません．ただし，信号用のリレーなどでは完全に接点が開閉し終わるまでの時間を応答時間として示しているものがあり，チャタリングはそれ以下だということがわかります．

チャタリングに不安をおもちの人は，ぜひチャタリングの実験をしてみることをおすすめします．これは図3.13のような簡単なものでもよくわかります．注意すべき点としてはつぎのようなものがあります．

(1) 個々のスイッチの間でチャタリング時間には数倍のバラツキがあることもある．少数の実測結果を過信しないこと．

(2) つき合わせ接点（コラムH参照）のオフの際にもチャタリングが見られる．

(3) すり合わせ接点では接点が移動している間，相当長い時間にわたってチャタリング（正確にいえば，これはチャタリングではないが）が見られる．

▶接点にはどれくらいの電圧・電流を印加すれば良いか　一般に，大電流用のスイッチを100μA以下の微小電流で使用しようとしても，接触が不安定で使いものになりません．

図3.13　チャタリングの実験回路

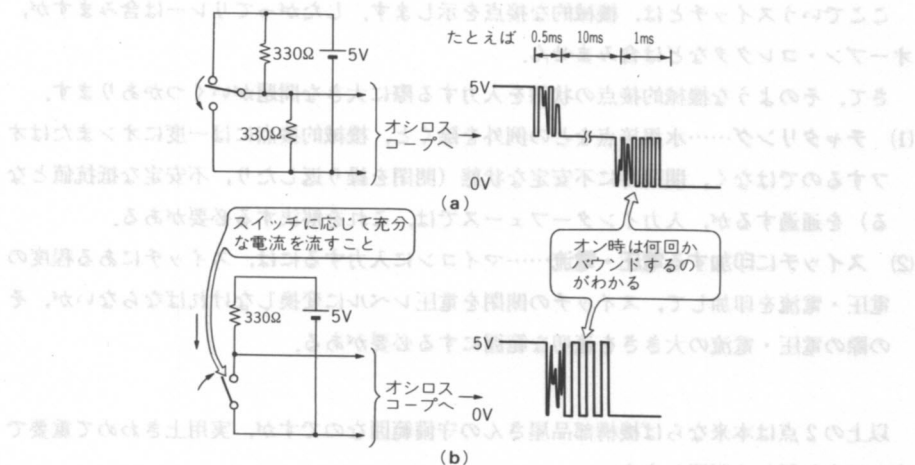
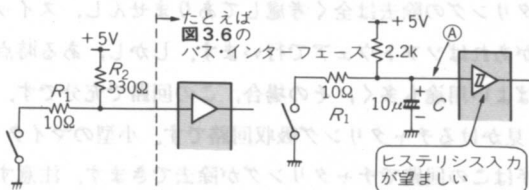



図 3.14 基本的なスイッチ用インターフェース

(a)単純にバスに接続した例 (b)チャタリング吸収回路付きの例

実際にパネルに取り付けられたり，リミット・スイッチとして用いられているスイッチを見ると，大電流用のトグル・スイッチやマイクロ・スイッチであることが少なくありません．“AC250V 10A”などという定格のスイッチには，少なくとも10mA 以上の電流を流したいものです．

一方，基板用のスイッチやキー・スイッチで微小電流用のものには，電流を流しすぎると寿命が短くなります．とくに，スイッチに直列に挿入する抵抗 (図 3.14 (b)の R_1) を省略したりすると，一時的に多くの電流が流れますので注意が必要です．

▶チャタリングの処理をハード/ソフトのいずれで行うか？ これは第2章でも触れましたが，ハード/ソフトそれぞれの除去能力は一長一短です．しかし，ソフトで行う場合はタイマ割込みでも使わないかぎり，貴重な時間を浪費していることになります．

また，重要でありながら意外と議論されていないのは，「本当にチャタリングを除去する必要があるのか否か？」という問題です．たとえば，オフの状態のスイッチを監視していて，オンになったら監視ループを抜ければよい場合では，雑音の問題を別にすればスイッチのチャタリングは問題となりません．

▶スイッチ入力インターフェース さて，前置きが長くなりましたがスイッチの入力方法の例を説明します．

図 3.14 (a)は例 3-1 (図 3.6) のバス・インターフェースにたんにスイッチを接続したものです．同一基板内で距離が 50mm くらいであれば $R_1 = 0$ でかまいませんが，スイッチまでの配線を延ばす場合には，$R_1 = 4.7 \sim 47\,\Omega$ を挿入します．これは配線のインダクタンスによる影響を軽減するためです．微小電流用でないスイッチを使用するなら，R_2には 10mA 以上の電流を流すようにします．基板用のスイッチには 1 mA 以下の電流でも安定なもの

が市販されています．

この回路はチャタリングの除去は全く考慮してありませんし，スイッチの状態も記憶しませんので，必要があればソフトウェアで行います．しかし，ある時点のスイッチの状態がわかりさえすればよい用途も多く，その場合，この回路で充分です．

図3.14(b)はよく見かけるチャタリング吸収回路です．小型のマイクロ・スイッチやトグル・スイッチの大半はこの回路でチャタリングが除去できます．注意すべき点はつぎのとおりです．

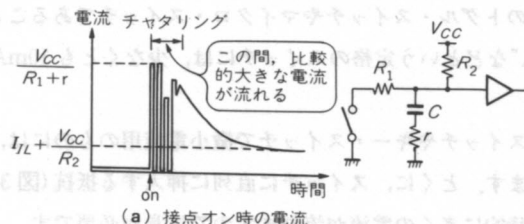
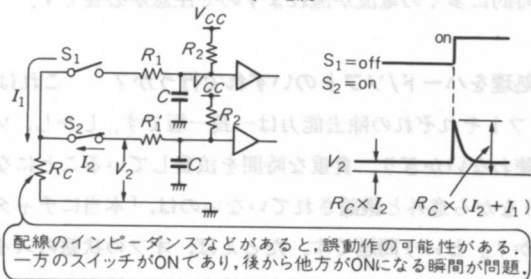
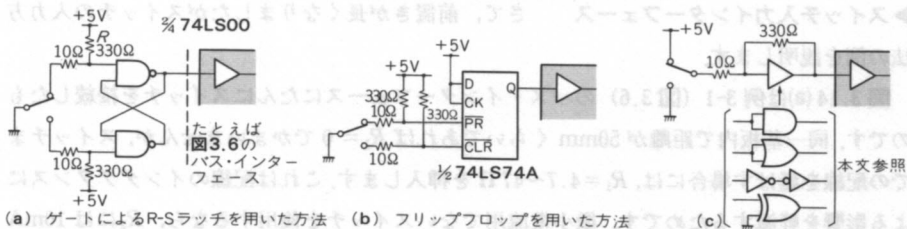
図3.15 接点に流れる電流による誤動作

(a) 接点オン時の電流

(b) 共通インピーダンスによる誤動作

図3.16 チャタリング除去回路――その1

(a) ゲートによるR-Sラッチを用いた方法 (b) フリップフロップを用いた方法 (c)

(1) スイッチを閉じた時点で $Vcc/(R_1+r)$ の電流が接点に流れる（r はコンデンサ C の内部抵抗および配線の抵抗）．とくに微小電流用のスイッチの定格オーバに注意．また，この電流によって他の回路に誤動作を生じさせないよう，配線にも注意すること(図3.15)．

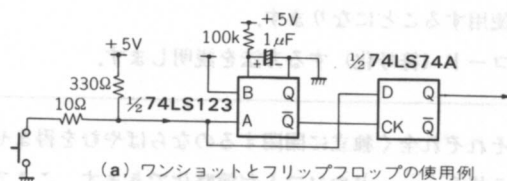
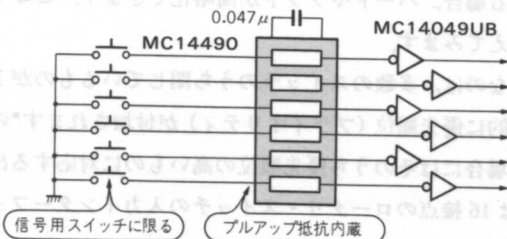
(2) 図3.14(b)Ⓐ点の信号は，立上りが遅いので，TTL で受けるのならヒステリシス入力のものがよい（74LS541 はヒステリシス入力）．

例 3-5 スイッチのチャタリング対策②

スイッチのチャタリングを除去するには他にも多くの方法があります．ここではその中からいくつかを紹介します．

前項でもっとも簡単なチャタリング除去方法を説明しましたので，それ以外の方法について述べておきます．

図3.16(a)はトランスファ接点をもつスイッチに適用できるものです．なお，トランスファ接点といっても，必ず BBM（ブレーク・ビフォア・メイク：切換えの際に一度断になる）動作でなければなりません．これは(b)，(c)の回路でも同様です．

図 3.17 チャタリング除去回路——その2

(a) ワンショットとフリップフロップの使用例

(b) 専用ICの使用例

図(a)ではゲートでRSラッチを組んでありますが，これを本物のフリップフロップにしても同様です（同図(b)）.

また，図(c)はフリップフロップと見てもよいし，極端にヒステリシスの大きいコンパレータと見ることもできます．スイッチに接続する配線が常に低インピーダンスとなり，雑音に強いことが特徴です．ここに使用する非反転のゲートはANDでもORでもかまいません．また，TTLではなくCMOSであっても使えます．

つぎに図3.17に示すのは，ソフトウェアでチャタリングを除くのと同様の考え方で，一定時間のタイマを設けるものです．

同図(a)はワンショット回路とDフリップフロップとを用いたもの，同図(b)は専用IC（MC 14490）を用いたものです．MC 14490の中身は発振器とカウンタとから成る一種のタイマで，IC 1個に6回路が内蔵されています．ただし，TTLは駆動できないのでバッファを必要とします．

例 3-6　多数のスイッチの入力①——エンコード

ロータリ・スイッチやテン・キー，あるいはフル・キーボードのように，スイッチの接点数が多い場合，以上に述べた方法だけでは配線の手間もさることながら，インターフェース回路の量が莫大となってしまいます．

そこでスイッチの種類や使用方法に応じて，入力インターフェースを簡単にするための様々な技法を使用することになります．

ここでは，エンコード（符号化）する方法を説明します．

多数のスイッチがそれぞれ全く独立に開閉するのならばやむを得ません．しかし，本章に示した方法が使える場合，ハードやソフトが簡略化できます．ここではエンコード（符号化）する方法を考えてみます．

エンコードが有効なのは，多数のスイッチのうち閉じているものが1個だけの場合です．エンコードには必然的に優先順位（プライオリティ）が付加されます*ので，2個以上のスイッチが閉じている場合にはそのうち優先順位の高いものに対応する出力が得られます．

図3.18に示すのは16接点のロータリ・スイッチの入力インターフェースです．同図(a)

* 押した順序まで含めて優先順位をつける（ロール・オーバ）こともある．

はエンコードしないで直接に入力する場合，同図(b)はエンコードした場合です．ハードウェア上の違いは，取り込むビット数が減ることぐらいですが，ソフトウェアでスイッチの接点位置を調べる手間がなくなることも考慮する必要があります．

ロータリ・スイッチの場合，接点位置が同時に2ヵ所以上に渡ることは（過渡状態を除いて）本質的にあり得ないわけですが，図3.19(a)に示すテン・キーなどの場合は生じ得ます．エンコード用のMSIには必ず優先順位がついていますので，テン・キーのうち2個以上のスイッチを押すと，数の大きいほうが出力されるようになります（逆順にすることも

図3.18　ロータリ・スイッチの入力

(a) エンコーダなし

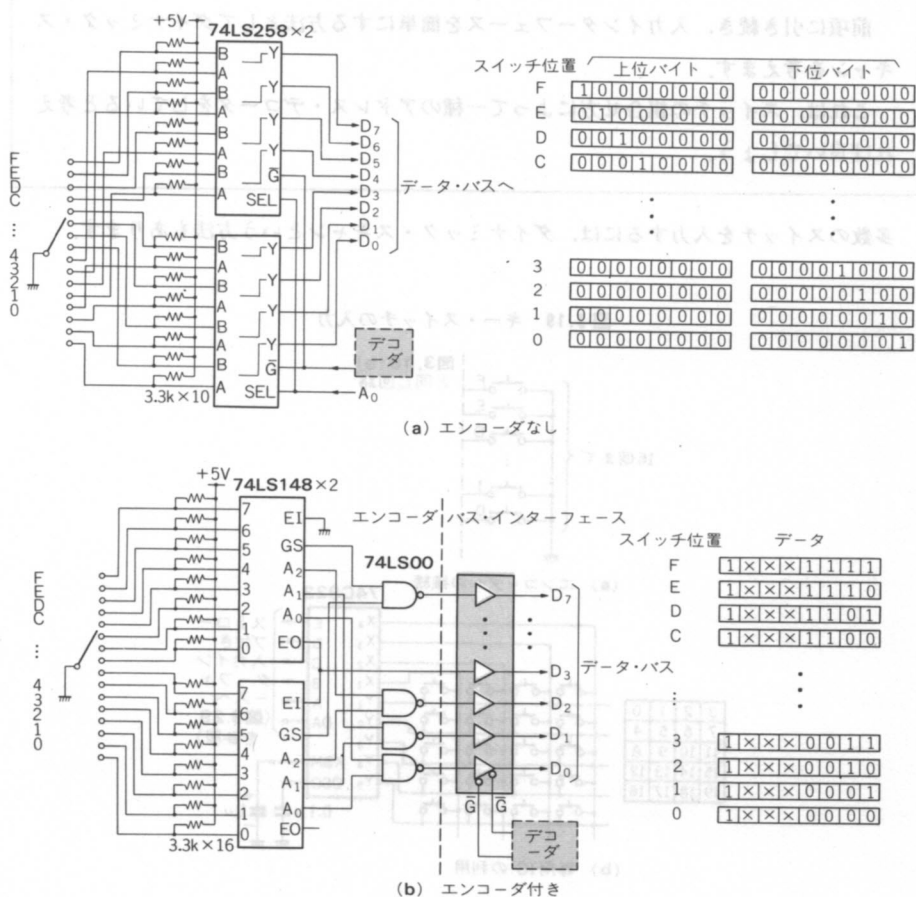
(b) エンコーダ付き

できる).

図3.19(b)の例は20個のスイッチのエンコード用の専用IC 74C923を用いた例です. このICは内部でダイナミック・スキャン(次項を参照)を行っているため, 20個のスイッチは4×5のマトリクス状に接続します. なお, 74C923の端子接続を図3.20に示します.

エンコード用のICとしては, フル・キー(50個以上のスイッチから成る)の専用エンコーダもあります.

例3-7　多数のスイッチの入力②――ダイナミック・スキャン

前項に引き続き, 入力インターフェースを簡単にする方法としてダイナミック・スキャンを考えます.

これは, スイッチの組合せ方によって一種のアドレス・デコーダをしていると考えれば良いでしょう.

多数のスイッチを入力するには, ダイナミック・スキャンという方法もあります.

図3.19　キー・スイッチの入力

(a) エンコーダへの接続

(b) 専用ICの利用

図 3.21 (a)に示すのは複数のロータリ・スイッチの接点位置を調べるもので，アドレスの下位ビットをデコードして1個のスイッチを選択します．図 3.21 (b)は複数のディジタル・スイッチ（サム・ホィール・スイッチ）を読み込むためのものです．ロータリ・スイッチと異なり，ディジタル・スイッチは2ヵ所以上のポジションが短絡されていることがあるのでダイオードで分離しなければなりません．

その際，ダイオードの順方向電圧によってTTLの$V_{IL} \leqq 0.8$Vを満足できなくなることがあるので，CMOSで受けます．また，この回路のようにアドレスの下位ビットをスキャンに使用する場合，必要なとき以外はデコーダ(74LS156)をアクティブにしないほうが雑

図 3.20 MM 74 C 923 の端子接続[10]

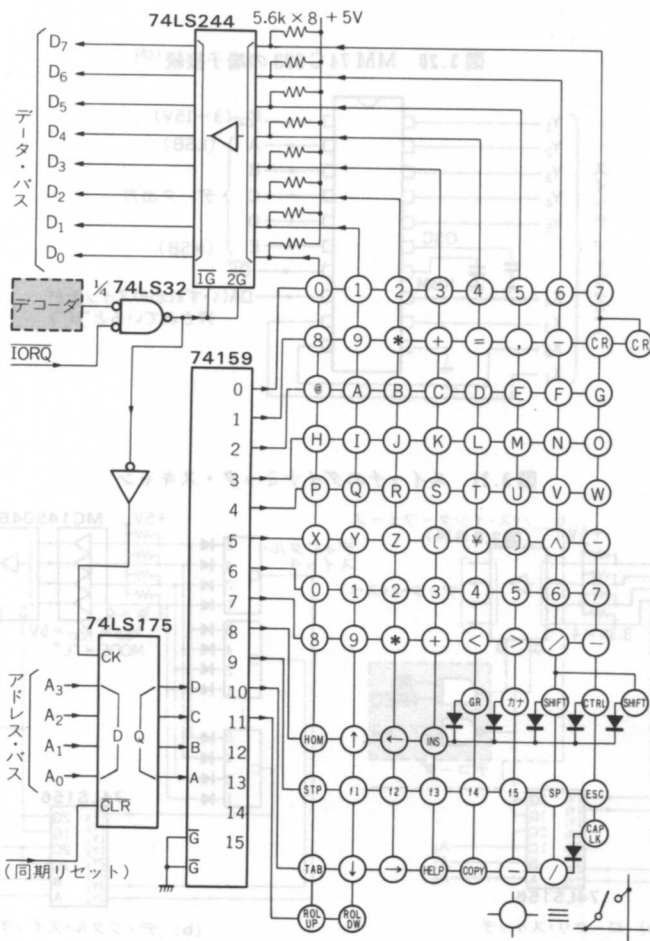
図 3.21 スイッチのダイナミック・スキャン

(a) ロータリ・スイッチ

(b) ディジタル・スイッチ

音などの点で望ましいといえます．

市販のパーソナル・コンピュータのうち価格帯で中以下のものの大部分は，キーボード入力にダイナミック・スキャンを用いてます．図3.22は日本電気のパーソナル・コンピュータPC-8801のキーボード入力回路です(日本電気から発表された回路ではない)．どのキーが押されたかの判断はソフトウェアで行っています．なお，このようにダイオードを通った信号をTTLで直接受けるのは，あまり感心できません．

図3.22　パーソナル・コンピュータのキーボード入力方法の例

例 3-8　オープン・コレクタ信号の入力

> シーケンサや測定器の出力には，リレー接点と並んでオープン・コレクタがよく使われています.
>
> スイッチ類のようなチャタリングや接点不良の問題がない反面，オープン・コレクタはトランジスタとしての電圧・電流上の制約があります.
>
> 本項ではオープン・コレクタ信号をバス・インターフェースに接続する方法を説明します.

オープン・コレクタ（FET 出力ならばオープン・ドレイン）出力はチャタリングの問題はなく，しかも接点と同様に任意の電圧レベルに変換容易な利点があり，またフォト・カプラを使えばアイソレーション（絶縁）が可能なこともあって，ディジタル信号の受け渡しに広く使われています.

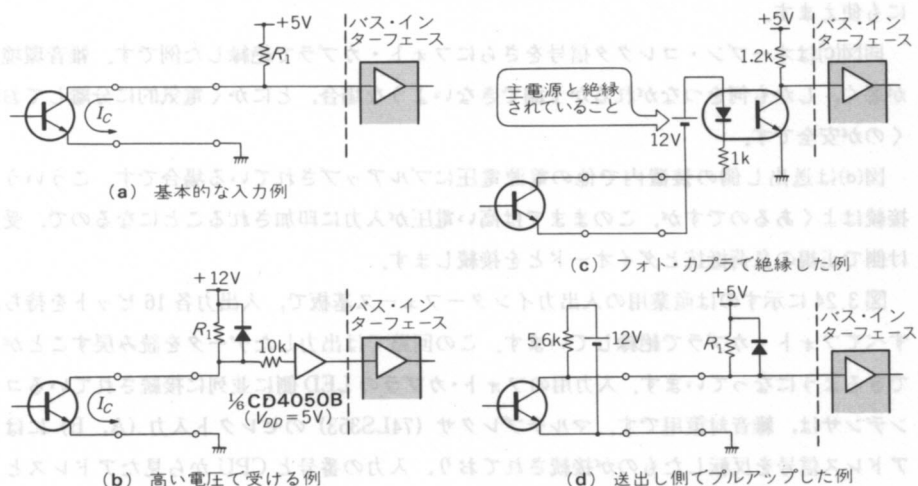
一方，オープン・コレクタは接点と違って飽和電圧が存在し，オフ時にも漏れ電流があります. また，耐圧やコレクタ電流の制限もあるので注意が必要です.

図 3.23　オープン・コレクタ信号の入力

(a)　基本的な入力例

(b)　高い電圧で受ける例

(c)　フォト・カプラで絶縁した例

(d)　送出し側でプルアップした例

▶オープン・コレクタの入力回路　図3.23(a)に示すのはオープン・コレクタ信号の基本的な入力方法です．ダーリントン接続でないオープン・コレクタ出力は，オン時の飽和電圧は0.3V程度以下になりますから，TTLやCMOSで直接受けることができます．プルアップ抵抗（オープン・コレクタの負荷の一部）はオフ時の漏れ電流から決定します．たとえば図(a)のようにTTLで受ける場合R_1はつぎのように選びます．

$$\frac{V_{CC}\ (\mathrm{max})}{I_C\ (\mathrm{max})} \leqq R_1 \leqq \frac{V_{CC}\ (\mathrm{min}) - V_{IH}\ (\mathrm{min})}{I_{IH}\ (\mathrm{max}) + I_{CE\,(\mathrm{OFF})}}$$

V_{CC}：電源電圧（5V±5%）
V_{IH} (min)：TTLの“H”レベル電圧の下限（2.4V）
I_{IH} (max)：TTLの“H”レベル流れ込み電流の最大値（40μA）
$I_{CE\,(\mathrm{OFF})}$：オープン・コレクタのオフ時の漏れ電流
I_C (max)：オープン・コレクタの最大許容コレクタ電流

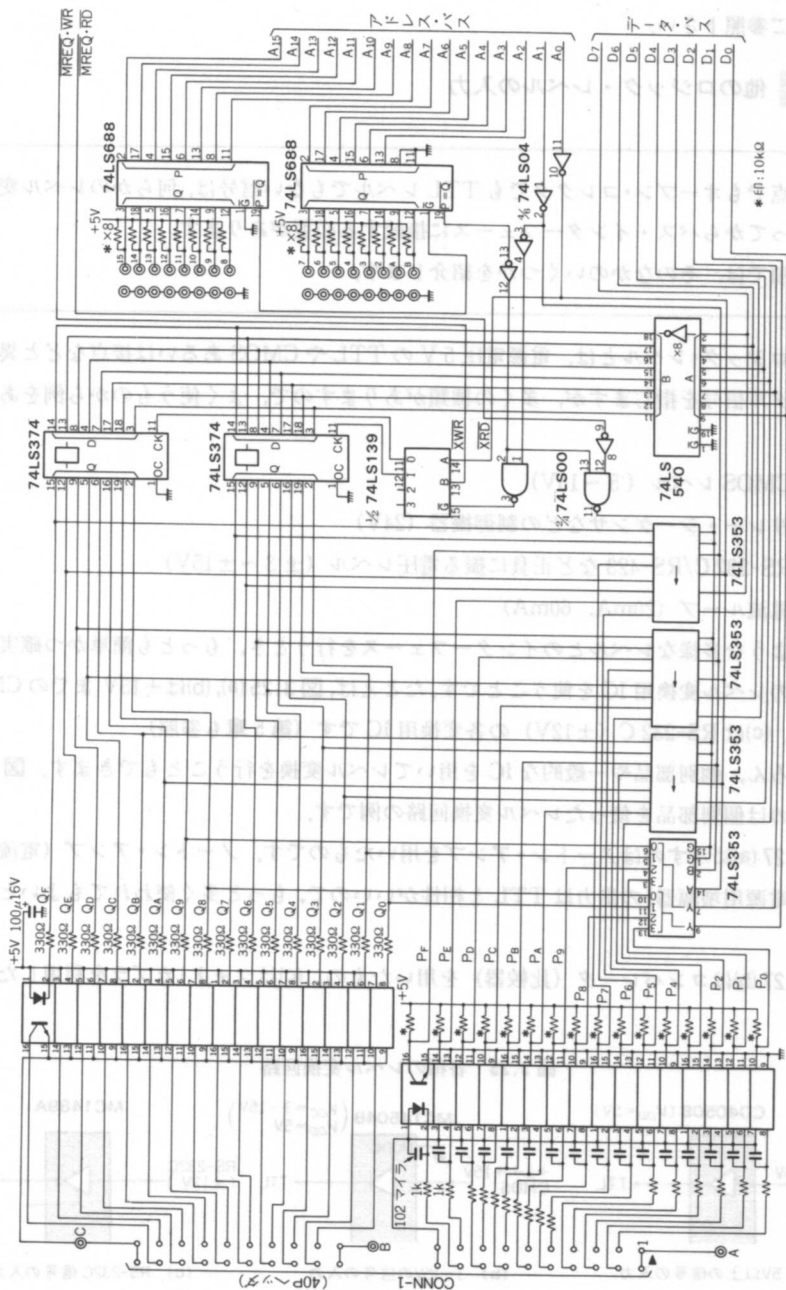
たとえば$I_{CE(\mathrm{OFF})}=100\mu\mathrm{A}$，$I_C\ (\mathrm{max})=10\mathrm{mA}$とすると，$525\Omega \leqq R_1 \leqq 16\mathrm{k}\Omega$となります．一般にオープン・コレクタが使用されるのは雑音環境のあまりよくない場所なので，R_1は低目にしたほうが安全です．

図3.23(b)はオープン・コレクタを高い電圧で受けるものです．これは耐雑音性などからレベルを高くしておきたい場合のほか，オープン・コレクタ側がダーリントン接続になっている場合（オン時の飽和電圧が0.7V以上になるため，TTLでは直接は受けられない）にも使えます．

同図(c)はオープン・コレクタ信号をさらにフォト・カプラで絶縁した例です．雑音環境が悪く，しかも何をつながれるか予測できないような場合，とにかく電気的に分離しておくのが安全です．

図(d)は送出し側の装置内で他の電源電圧にプルアップされている場合です．こういう接続はよくあるのですが，このままでは高い電圧が入力に印加されることになるので，受け側で正規の負荷抵抗とダイオードとを接続します．

図3.24に示すのは産業用の入出力インターフェース基板で，入出力各16ビットを持ち，すべてフォト・カプラで絶縁しています．この回路では出力したデータを読み戻すことができるようになっています．入力用のフォト・カプラのLED側に並列に接続されているコンデンサは，雑音対策用です．マルチプレクサ（74LS353）のセレクト入力（A，B）にはアドレス信号を反転したものが接続されており，入力の番号とCPUから見たアドレスと逆順になっていることに注意して下さい．なお，出力インターフェース(例4-1，例4-9な

図 3.24 入出力インターフェース基板

ど）もご参照下さい.

例 3-9　他のロジック・レベルの入力

> 接点でもオープン・コレクタでもTTLレベルでもない信号は，何らかのレベル変換を行ってからバス・インターフェースに接続する必要があります．
> 本項では，そのなかのいくつかを紹介します．

他のロジック・レベルとは，電源電圧5VのTTLやCMOSあるいは接点などと異なるディジタル信号を指しますが，多くの種類がありますので，よく使うものから例をあげてみます．

(1)　CMOSレベル（5～15V）

(2)　リレー・シーケンサなどの制御機器（24V）

(3)　RS-232C/RS-423など正負に振る電圧レベル（±3～±15V）

(4)　電流ループ（20mA，60mA）

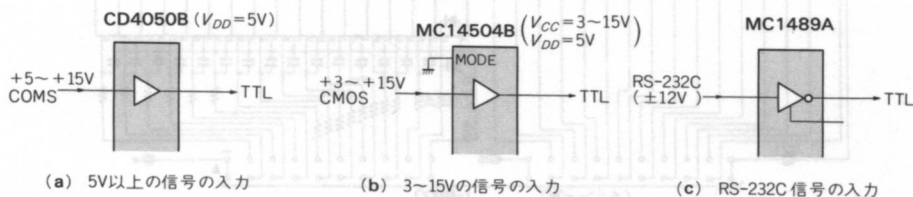
このような多様なレベルとのインターフェースを行うとき，もっとも簡単かつ確実なのは専用のレベル変換用ICを使うことです．たとえば，図3.25(a)，(b)は+15VまでのCMOSレベル，(c)はRS-232C（±12V）の各変換用ICです（第5章も参照）．

もちろん，個別部品や一般的なICを用いてレベル変換を行うこともできます．図3.26に示すのは個別部品を使ったレベル変換回路の例です．

図3.27(a)に示すのはノートン・アンプを用いたものです．ノートン・アンプ（電流入力の単一電源用増幅器）の出力はTTLと相性がいいので，もっと多く使われてもよいと思います．

図3.27(b)はコンパレータ（比較器）を用いたもの，(c)はフォト・カプラを利用したもの

図 3.25　各種のレベル変換回路

(a)　5V以上の信号の入力　　(b)　3～15Vの信号の入力　　(c)　RS-232C信号の入力

です．フォト・カプラは絶縁とともにレベル変換が可能です．同図(d)はCMOSのコンパレータですが，入力電圧を（電源電圧－2V）以下に抑えねばなりません．

一般に，専用のレベル変換用ICに比べて，個別部品などで組んだレベル変換回路は速度が遅くなりがちなので注意が必要です．

図 3.26 個別部品によるレベル変換回路

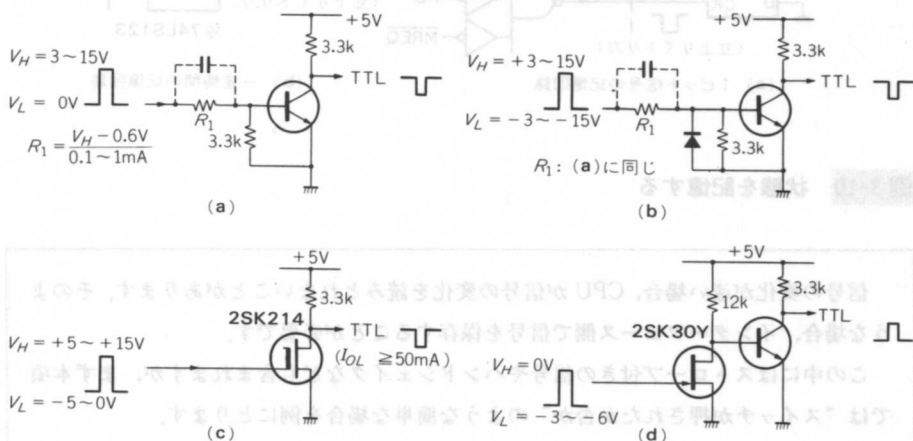
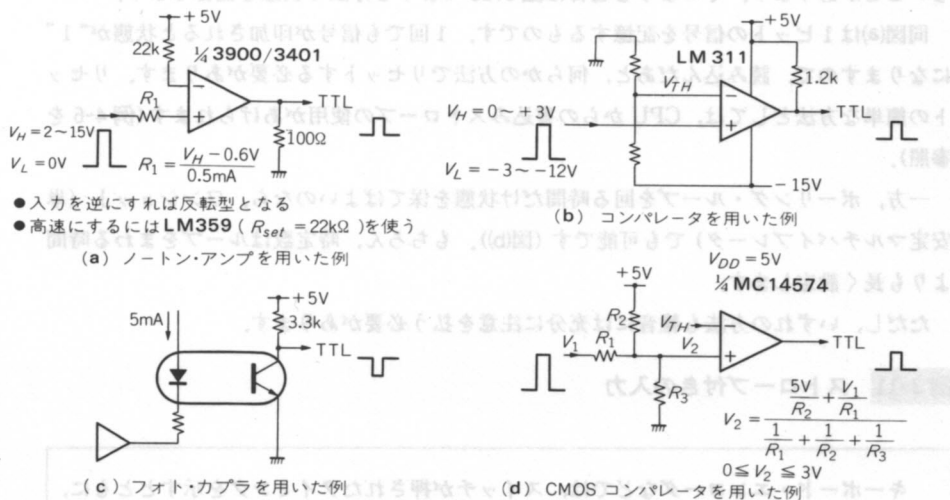
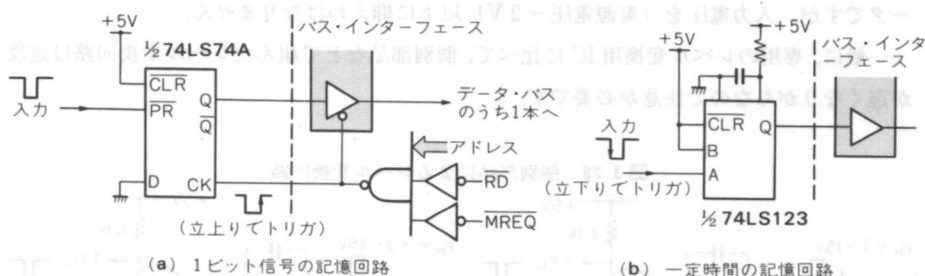



図 3.27 その他のレベル変換回路

図3.28 状態を記憶する

(a) 1ビット信号の記憶回路 (b) 一定時間の記憶回路

例3-10 状態を記憶する

> 信号の変化が速い場合，CPUが信号の変化を読みとれないことがあります．そのような場合，インターフェース側で信号を保存することが必要です．
>
> この中にはストローブ付きの信号やハンドシェイクなども含まれますが，まず本項では“スイッチが押されたか否か”のような簡単な場合を例にとります．

たとえば，信号が1回印加されたら，つぎにそれを読み込むまで記憶しておかねばならないことがあります．そのような場合は図3.28のような方法で状態を記憶します．

同図(a)は1ビットの信号を記憶するものです．1回でも信号が印加されると状態が“1”になりますので，読み込んだあと，何らかの方法でリセットする必要があります．リセットの簡単な方法としては，CPUからの書込みストローブの使用があげられます(例4-6を参照)．

一方，ポーリング・ループを回る時間だけ状態を保てばよいのなら，ワンショット（単安定マルチバイブレータ）でも可能です（図(b)）．もちろん，時定数はループをまわる時間よりも長く設定します．

ただし，いずれの方法も雑音には充分に注意を払う必要があります．

例3-11 ストローブ付きの入力

> キーボード・エンコーダなどでは，スイッチが押されたタイミングを示すとともに，

データが有効となったことを示すためストローブ信号が出力されます．また，測定器などでは信号を1桁分ずつ時分割で出力するとともに，それが何桁目のデータかを示すストローブ信号が付加されていることがあります．

本項では，このようなストローブ信号の使い方を示します．

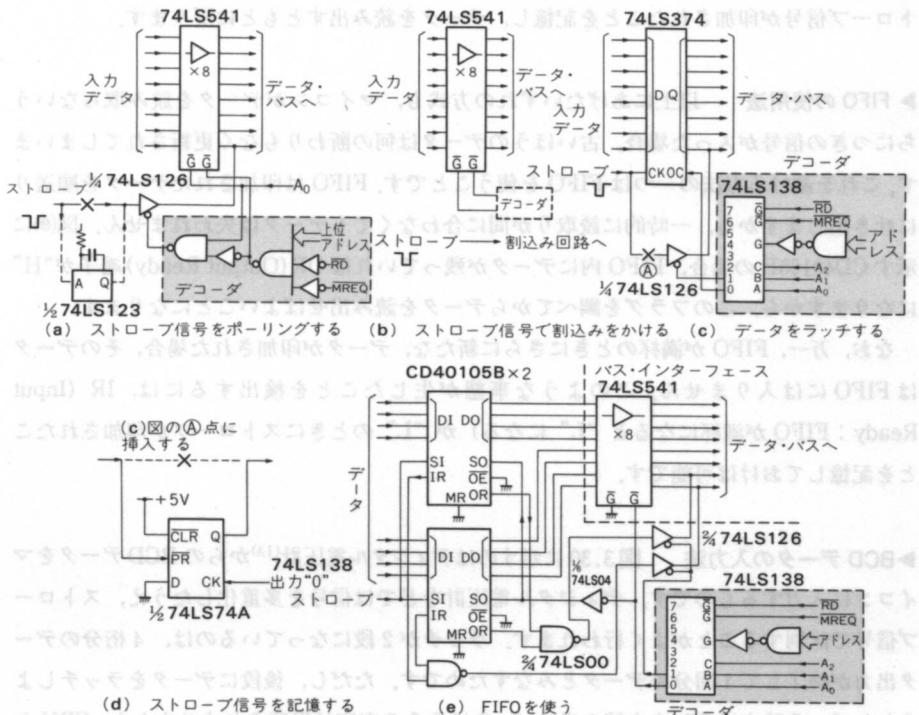
▶キーボード回路 エンコーダ付きのキーボードなどでは，キーを押すとエンコードされた値とともに，データが有効であることを示すストローブ信号が出力されます．

例3-6の図3-18(b)に示したエンコード回路では，74LS148のGS出力がこのストローブ信号に相当しますが，これは1個以上のスイッチを押している間はずっとアクティブになったままです．

これに対して，フル・キーボードなどに使われているエンコーダは，スイッチを押した

図3.29 ストローブ付きの入力

(a) ストローブ信号をポーリングする (b) ストローブ信号で割込みをかける (c) データをラッチする

(d) ストローブ信号を記憶する (e) FIFOを使う

あと所定の幅のパルス状のストローブ信号を出力します．このパルス幅は1～100μsのものが多いようです．

▶**ストローブ入力信号の処理方法**　　さて，マイコン側がストローブ信号のポーリングに専念できるならば，100μsのストローブ・パルス幅を読み取ることは可能です．ストローブ・パルス幅が短かければ，ワンショットで100μs程度まで延ばす手があります（図3.29(a)）．

マイコンが充分な速さでのポーリングを行えない場合，もっとも有効なのはストローブ信号で割込みをかけることです(図(b))．ただし，この方法も図(a)に示した方法も，ストローブ信号がアクティブでなくなってからもデータが保持される場合にしか使用できません．

確実な動作を望むのならば，図3.29(c)，(d)のような回路を使用します．いずれもストローブ信号を用いてデータをラッチするものですが，(c)は充分速いポーリングまたは割込みを使用するもの，(d)はそうではない場合に使用できるものです．すなわち(d)の回路は，ストローブ信号が印加されたことを記憶し，データを読み出すともとに戻ります．

▶**FIFOの使用法**　　以上にあげたいずれの方式も，マイコンがデータを読み取らないうちにつぎの信号が入った場合，古いほうのデータは何の断わりもなく更新されてしまいます．これを避ける方法の一つはFIFOを使うことです．FIFOは印加されたデータを順送りに吐き出しますから，一時的に読取りが間に合わなくてもデータは失われません．図(e)に示すCD40105Bの場合，FIFO内にデータが残っていればOR(Output Ready)端子が“H”になりますから，このフラグを調べてからデータを読み出せばよいことになります．

なお，万一，FIFOが満杯のときにさらに新たな，データが印加された場合，そのデータはFIFOには入りません．このような事態が生じたことを検出するには，IR（Input Ready：FIFOが満杯になると“L”になる）が“L”のときにストローブが印加されたことを記憶しておけば可能です．

▶**BCDデータの入力法**　　図3.30に示すのはディジタル電圧計[13]からのBCDデータをマイコンに入力するものです．ディジタル電圧計などでは信号を多重化したうえ，ストローブ信号で区別することがよく行われます．ラッチが2段になっているのは，4桁分のデータ出力が終了して1回分のデータとみなすためです．ただし，後段にデータをラッチしようとしている時点でデータを読み出せば，やはりその内容は保証されませんから，CPUか

らのデータ読出しは必ずステータス（図3.30ではEOC信号）を調べてからにします．

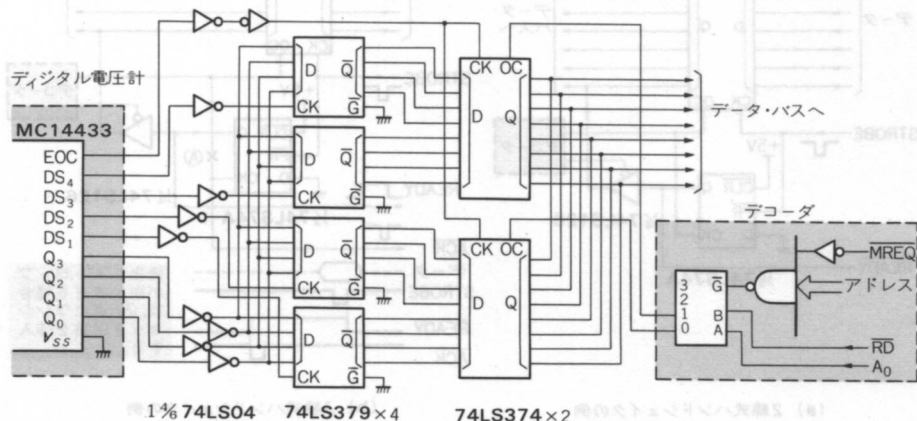
例3-12　ハンドシェイク入力

確実な信号転送を行うには，送り出し側が次の信号を出力する前に，受け取り側が前の信号を入力したのを確かめる必要があります．この確認操作がハンドシェイクです．

ハンドシェイクはプリンタとのインターフェース（セントロニクス型ハンドシェイク）やGPIBなど，マイコンと周辺機器とのインターフェースに幅広く使われています．なお，第4章のコラムJも参照してください．

データ転送の際に，転送が確実に行われたことを確認しながら行うには，ハンドシェイクを用います．

図3.31に示すのは基本的なハンドシェイクの構成で，送り出し側がデータを出力するとともに，フリップフロップをセットします．受け取り側はフリップフロップの状態を調べて，セットされていればデータを読み込むとともにフリップフロップをリセットします．送り出し側が新たにデータを出力する場合，フリップフロップがリセットされていることを確認してから行います．このようなハンドシェイクを行えば，データが消失することはありません．

図3.30　ディジタル電圧計からのデータ読込み

▶ハンドシェイクの回路例　図3.32に示すのはハンドシェイク型入力インターフェースの例です．

図(a)は基本的なもので，ストローブ信号でフリップフロップをセットし，読出しと同時にリセットします．このような方式を2線式ハンドシェイクと呼びます．なお，データの送り出し側がハンドシェイク終了までデータを保持している場合はデータをラッチする必

図3.31　基本的なハンドシェイク

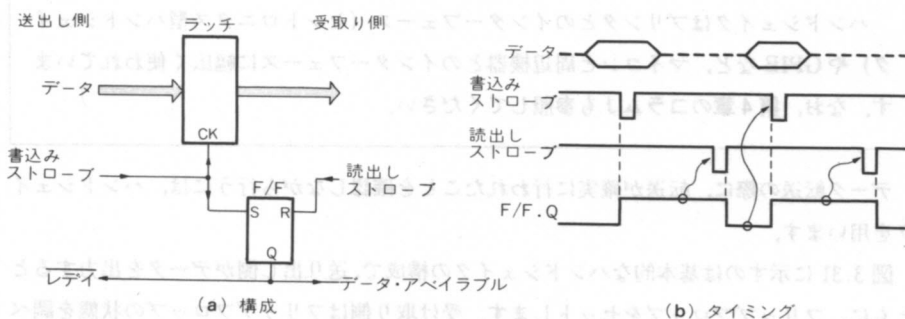
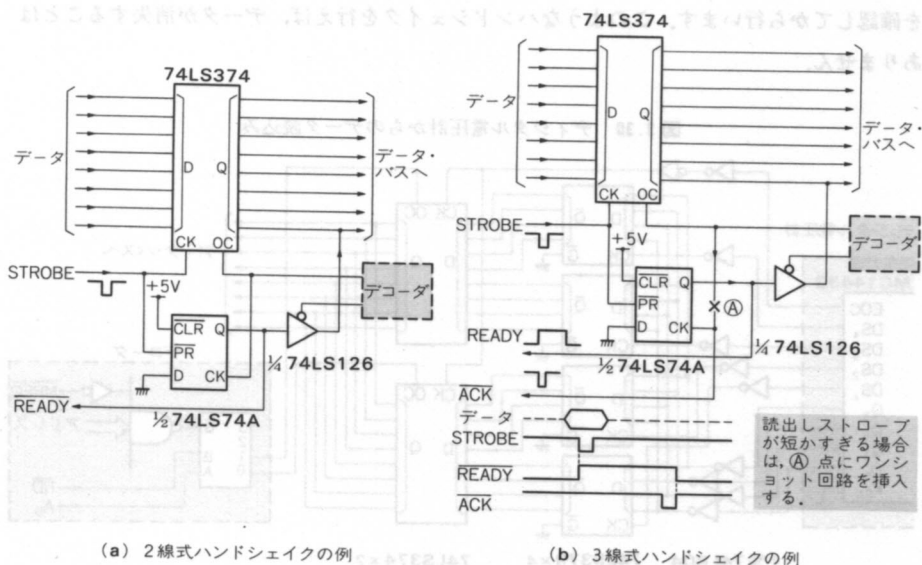
送出し側　ラッチ　受取り側
データ
CK
書込み
ストローブ
F/F
S　R
Q
読出し
ストローブ
レディ
データ・アベイラブル

(a)　構成

データ
書込み
ストローブ
読出し
ストローブ
F/F.Q

(b)　タイミング

図3.32　ハンドシェイクを用いた入力

(a)　2線式ハンドシェイクの例　　　(b)　3線式ハンドシェイクの例

要はなく，74LS374のかわりに74LS541などのスリーステート・バッファでも使えます．

図(b)は3線式ハンドシェイクです．GPIB（IEEE-488）と呼ばれるインターフェースは，基本的にはこの3線式ハンドシェイクです．

メーカ製のパーソナル・コンピュータにはプリンタ用の出力ポート（多くはセントロニクス型と称している）が付いており，BASICのPRINT文で簡単にデータを出力できます．これに対して図3.31，図3.32のような入力インターフェースをもっていれば，他のマイコンからのデータをもっとも簡単に受け取ることができます．

なお，ハンドシェイク出力の方法については例4-7も参照してください．

例3-13 基本的なアナログ入力①——レベル検出

> マイコンにアナログ信号を入力する場合，一般的にはA-D変換器を使用しますが，“所定のレベルを越えたか否か”を知りたい場合はレベル・コンパレータ（比較器：1ビットのA-D変換器と見てもよい）が使用できます．
>
> ここではその例と，基本的な注意事項とを示します．

アナログ信号といっても多くの種類がありますが，ここでは，信号のレベルや形態はA-D変換器の入力に適合するように処理されているものとします．

さて，アナログ信号をマイコンに取り込むにはA-D変換を行わねばなりません．アナログ入力インターフェースの設計の中では，A-D変換器を選択することがもっとも重要だといえるでしょう．

従来，A-D変換器は高速の逐次比較型と高分解能の2重積分型とが代表的といわれてきました．しかし，最近になって従来の欠点を改良した高速･高分解能のA-D変換器が数多く発表されています．また，マイコンが必要とするA-D変換器は，従来の計測用のものに比べて速度・分解能ともに多少異なったものとなっています．したがって，本書で述べるA-D変換器の分類は従来のものとは必ずしも一致しません．

▶レベル検出回路　図3.33に示すのは簡単なレベル検出回路です．A-D変換器のもっとも簡単なものは，このようなコンパレータ（比較器）です．レベル検出回路の設計には，つぎのような点に注意します．

(1) 入力レベルに注意する．たとえばLM319は，＋5V（単一）にまで電源電圧を下げ

て使うことができ，ディジタル回路との共存に便利だが，＋5Vで使ったときの入力電圧範囲は＋1～＋3Vに過ぎない．

⑵ 必要に応じてヒステリシス（正帰還）を考慮する．これは遅い信号変化の際のマルチプル・トリガを防止するためと考えたらよい．上側・下側の二つの離れたしきい値をもたせたいのであれば，コンパレータを2個使ったほうが簡単である．

▶応答速度の問題　このほかコンパレータの応答速度の問題がありますが，2901系のような比較的低速のものでも1～2μsで，ソフトウェアで読み込む速度に比べれば，1桁は高速です．

さらに高速なコンパレータが必要ならば，図3.34に示すμA760（25ns（max））のような高速のものを用います．また，SN75207などのライン・レシーバやセンス・アンプは一般に高速で，しかも出力がTTLレベルとなっているので意外に便利です．欠点は，入力バイアス電流やオフセット電圧が本物（？）のコンパレータよりも大きいことです．

図3.33　コンパレータを用いたレベル検出

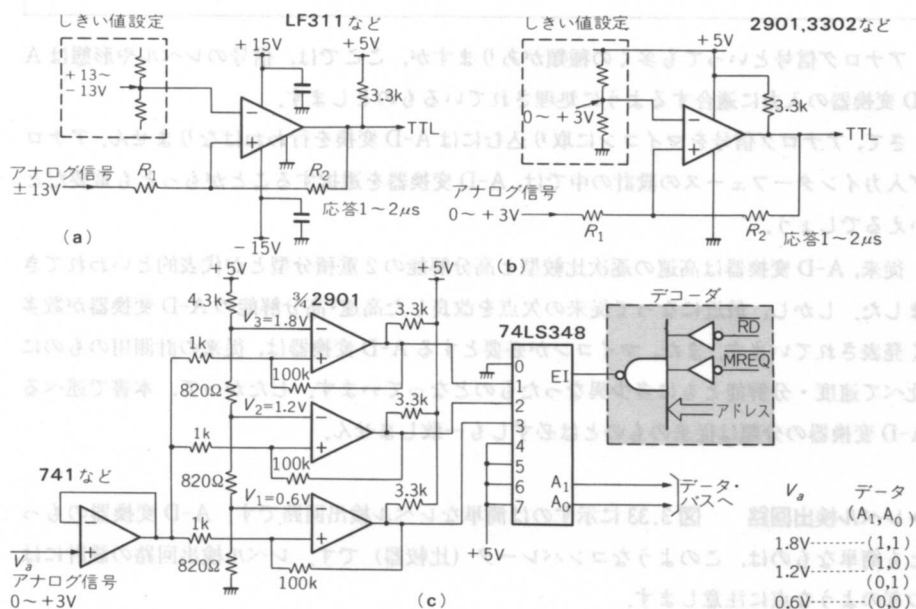
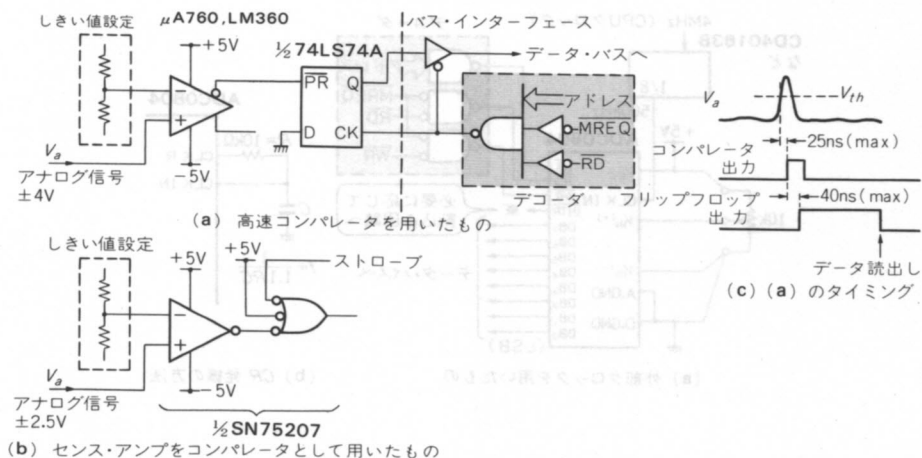



図3.34 高速のパルス検出

例3-14　基本的なアナログ入力②――8ビット中速A-D変換器

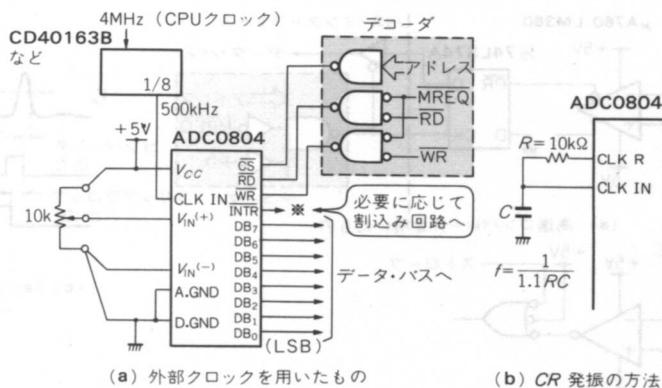
> アナログ信号を1%程度の精度や分解能で取り込みたい場合，8ビットのA-D変換器が便利です．これはマイコンのデータ・バス幅に一致しており，処理も簡単です．
> 本項では，このようなA-D変換器の使い方を紹介します．

センサやトランスジューサからの簡単な信号の取込み，ボリュームやポテンショメータの位置の読込みなどのうち8ビット（1バイト）以下の分解能ですむのであれば，安価なA-D変換器が数多くあります．

図3.35に示すのはNS社の8ビットA-D変換器ADC 0801シリーズ[8]です．このICはセカンド・ソースも多く，小型で安価です．このICのA-D変換方式は逐次比較方式なのですが，従来使われてきたR-$2R$ラダーではなく，リニアな分圧回路を用い，内部のコンパレータもダイナミック型とするなど，新しい技術が用いられています．ディジタル部分はCMOS構造として消費電力を抑えています．

▶ A-D変換回路例　図3-35(a)は5Vの電源に接続されたポテンショメータの位置を読み込む例です．このICの入力電圧範囲は原則として0～5V（電源電圧以内）であり，し

図3.35　8ビット中速のA-D変換

（a）外部クロックを用いたもの　　（b）CR発振の方法

かも電源電圧を基準電圧源として使用します．したがってポテンショメータのように電源電圧に対する分圧比を求めるには最適といえます．

変換にはクロック信号が必要で，同図(a)のようにマイコン側のクロックを流用してもよいし，(b)のようにCRで発振させることもできます．外部からクロックを印加する場合，デューティ比が50％(±10％)となるように注意します．クロック周波数と変換時間との関係は，

変換時間＝（1/クロック周波数）×(66〜73)

となります．精度を悪くしない範囲では，クロック周波数640kHz，変換時間約110μsが上限です．

▶マイコン・バス側　つぎにマイコン・バス側を見てみましょう．このICは$\overline{\mathrm{CS}}$と$\overline{\mathrm{WR}}$とをともに“L”にすると，そのいずれかの立上りから変換動作を開始します．このときのデータ・バスの信号は何でもかまいません．変換を開始すると$\overline{\mathrm{INTR}}$端子が“H”になり，終了すると“L”になりますから，この状態を割込みやポーリングで検出するか，あるいは必要な変換時間だけ待機して，データを読み出します．読出しは$\overline{\mathrm{CS}}$と$\overline{\mathrm{RD}}$とをともに“L”にします．アクセス時間は最大で200ns(C_L=100pF)ですから，4MHzのZ80Aにはメモリ領域，I/O領域のいずれにも直結できます(図2.9参照)．直流的にはV_{OL}=0.4V(I_{OL}=1.6mA)です．

なお，このA-D変換器のアナログ入力端子のインピーダンスは内部のスイッチングに

より変動しますので，ポテンショメータまでの距離が長い場合は注意が必要です．

例 3-15 高速のアナログ入力——高速 A-D 変換上の問題点

アナログ信号もその変化が高速になるとマイコンでの処理はしだいに困難になります．そのため信号処理専用のハードウェアも出てきています．しかし，音声帯域までならばマイコンでも何とか取り扱えます．

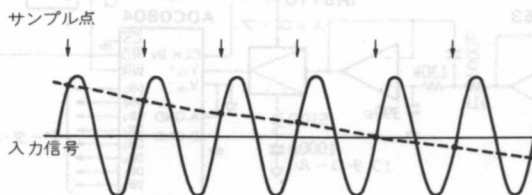
本項で説明するのは変化するアナログ信号をマイコンで入力する際の基本についてです．

高速といっても，ここでは変換周期 30〜300μs くらいを考えます．これ以上の速度になるとソフトウェアによる読込みは困難で，DMA を考慮せざるを得ません．この速度であってもマイコン側は一定数のデータを取り込む間はそれに専念することが必要です．

一般に，短い変換周期を必要とするのは，入力信号もそれに応じた変化をともなう場合です．ところが，積分型は別として，逐次比較型などの A-D 変換器は，原則として変換中に入力信号を変化させると，正しい A-D 変換ができません．また，変換周期に近い周波数の信号を入力すると，折返し歪が現れます（図 3.36）．

▶ローパス・フィルタとサンプル・ホールド回路　このようなことから，高速の A-D 変換を行うには入力信号の帯域に応じて，ローパス・フィルタやサンプル・アンド・ホールド回路を付加する必要があります．

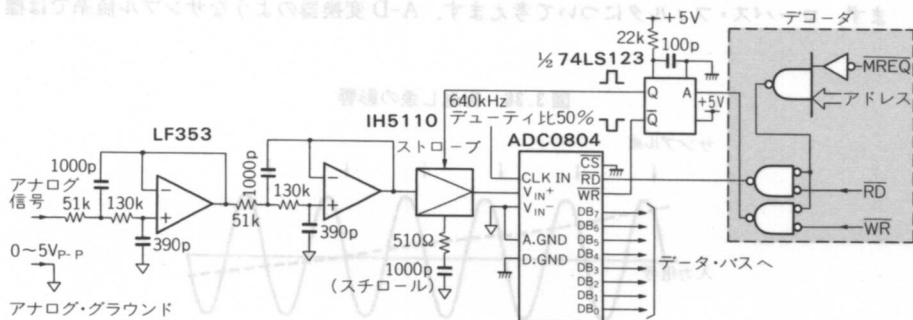
まず，ローパス・フィルタについて考えます．A-D 変換器のようなサンプル値系では標

図 3.36 折返し歪の影響

本化定理からサンプル周波数の1/2の帯域までしか正しい情報として扱えません．また，折返し歪の点でもっとも問題となるサンプル周波数付近では，必要なビット数に応じて充分に信号を減衰させることが必要です．たとえば，フィルタのカットオフ周波数をサンプル周波数の1/4とし，サンプル周波数付近で60dB減衰がほしいとすれば，30dB/octの傾斜が必要となります．

つぎにサンプル・アンド・ホールド回路は，必要な精度（たとえば1/2 LSB以下）で必要なアクイジション時間およびドループ率をもたねばなりません．

▶高速A-D変換回路　図3.37に示すのは以上を考慮したアナログ信号入力インターフェースの1例です．アナログ信号はカットオフ周波数3 kHz，24dB/octのローパス・フィルタと，サンプル・アンド・ホールド回路とを通ってA-D変換器に印加されます．ここで用いたサンプル・アンド・ホールド回路(インターシル社IH 5110)[7]は出力の立上りの制限からアクイジション時間があまり短くできません．このため，CPUからの書込みストローブ信号を延長して使用しています．ドループ率はA-D変換時間内に1/2 LSBを目安とします．

この例では分解能も8ビット，変換周期も100μs以上ですが，それでも簡易型として音声信号などをマイコンに取り込むことができます．もっとも，汎用のマイコンでは毎秒8 Kバイトで取り込んだデータを処理しようとすると，簡単なものでもかなりの時間を必要とするでしょう．

図3.37　8 K サンプル/秒，8ビットのアナログ入力インターフェース

例 3-16　高分解能アナログ入力――必要な精度と分解能を得る方法

工業用の測定では遅くてもよいかわりに10ビット以上の分解能が必要なことがあります.

しかし, 高い分解能や精度が必要な場合, たんに所定ビット数のA-D変換器を接続するだけでなく, いくつかの注意が必要となります. 本項では, そのような場合の基礎事項を説明します.

工業計測用としては10ビット以上のA-D変換が必要とされることがあります. ビット数と分解能との関係を表3.1に示しますが, 符号込みの12ビットはだいたい3½桁に相当します.

▶分解能と精度　ここでひとつ注意しておきますが, 一般に分解能と精度（広義の）とは異なります. つまり, 分解能12ビットで精度（たとえば直線性誤差）0.1％ということはあり得ます.

また, A-D変換器には必ず量子化誤差（±1/2ビットに相当する）が存在します（図3.

表3.1　分解能・ビット数と量子化誤差

分解能（符号込み）	フルスケールのディジタル値		量子化誤差
	符号なし	符号あり	
6ビット	0～　63	−32～＋31	0.78 ％
7	0～　127	−64～＋63	0.39 ％
8	0～　255	−128～＋127	0.20 ％
9	0～　511	−256～＋255	0.10 ％
10	0～ 1023	−512～＋511	0.049％
11	0～ 2047	−1024～＋1023	0.024％
12	0～ 4095	−2048～＋2047	0.012％
13	0～ 8191	−4096～＋4095	61 ppm
14	0～16383	−8192～＋8191	31 ppm
15	0～32767	−16384～＋16383	15 ppm
16	0～65535	−32768～＋32767	7.6ppm

38(a))．A-D変換器の誤差の表記にはいろいろな方法がありますが，全体の精度を求めるには表記されている誤差のほかに量子化誤差を考えることを忘れないようにします．

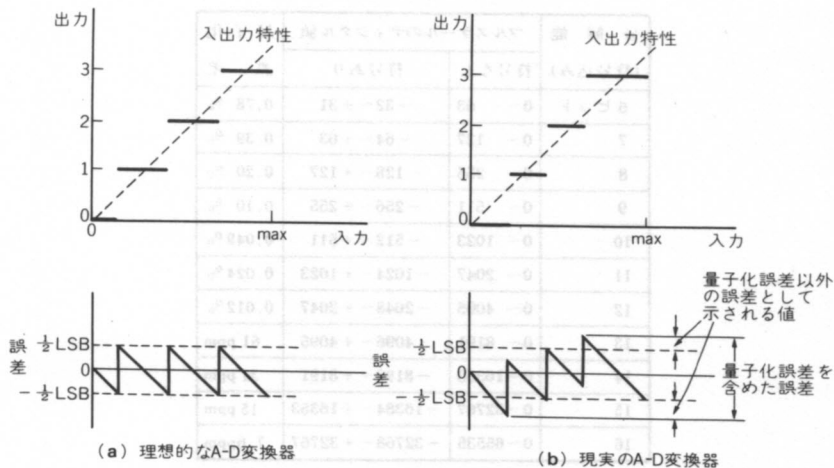
▶高分解能A-D変換回路　さて，10ビット以上の分解能を必要とするアナログ信号の入力方法を図3.39に示します．

図3.39(a)はインターシル社のICL7109を用いたものです．このA-D変換器は極性＋12ビットの分解能をもち，入力範囲は±0.4V（分解能10μV/ビット）以上で任意に設定でき，出力がスリーステートになるなど，マイコン用として多くの利点をもっています．A-D変換は基本的には2重積分型で，毎秒30回くらいが限界ですので低速の部類に入りますが，マイコンを用いた工業用計測には幅広く使えると思います．

このA-D変換器のRUN/$\overline{\text{HOLD}}$端子は“H”にすれば変換を開始し，“L”にすれば続行中の変換を終了してから停止します．“H”に固定しておけば連続して変換を続けます．

変換を終わるとSTATUS端子が“H”から“L”になりますので，これを割込みまたはポーリングで検査してデータを読み込みます．出力は図3.39(b)のようなビット構成となっています．ICL7109自体もバス・インターフェースを内蔵しており，そのアクセス時間は$\overline{\text{CE}}$から400ns，$\overline{\text{HBEN}}$または$\overline{\text{LEBN}}$からは各350nsとなっているので4MHzのZ80AのI/O領域への直結も可能です(図3.39(c))．ただし，バス負荷が増した場合の遅れの度

図3.38　A-D変換器の誤差

（a）理想的なA-D変換器　　（b）現実のA-D変換器

合いが明示されていない点と，出力が非アクティブ(スリーステート)に戻るまでの時間が長いためバス衝突の可能性がある点とを考えると，バッファを付加したほうが安全です.

▶必要な精度を得るためには　このように分解能が12ビット以上であっても，マイコンに接続すること自体は簡単です．ただし必要な精度を得ることはけっして簡単ではありません．詳しいことは他書にゆずるとして，ここでは二つの項目を示しておきます.

(1) **アナログ・グラウンドとディジタル・グラウンドとは分離されているか.**

マイコン自体は巨大なディジタル回路であり，発生する雑音はけっして少なくありませ

図 3.39　12ビット低速 A-D 変換器

(a)

(b) バス・インターフェースとビット配置　　(c) バスに直接接続する

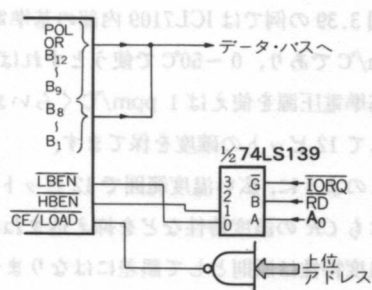
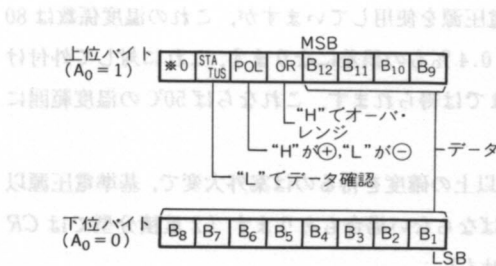
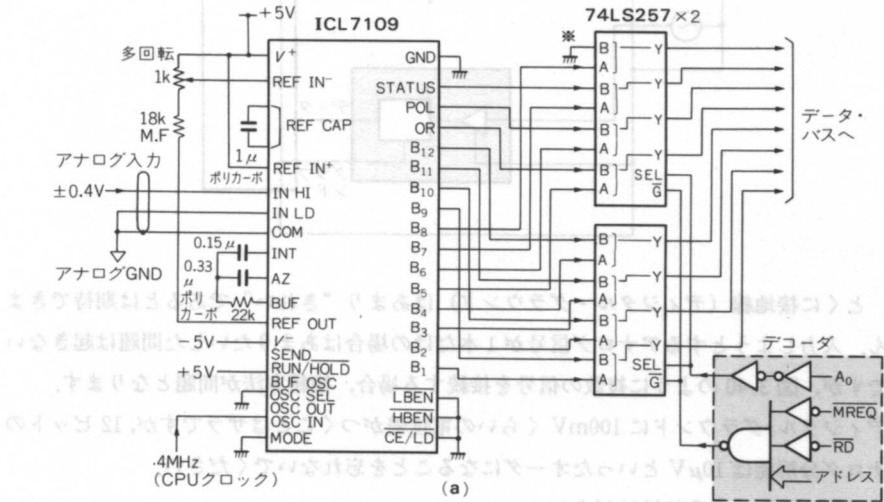
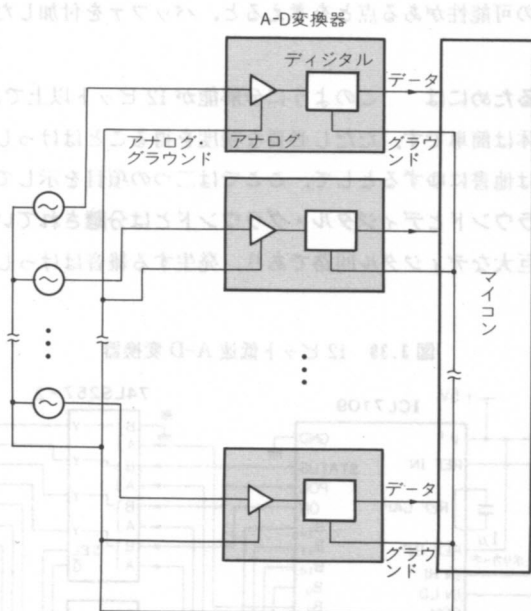



図 3.40　アナログ・グラウンドの接続

ん．とくに接地線（ディジタル・グラウンド）はあまり“きれい”であるとは期待できません．入力しようとするアナログ信号が1本だけの場合はあまりたいした問題は起きないのですが，図3.40のように複数の信号を接続する場合，接続方法が問題となります．

ディジタル・グラウンドに100mVくらいの電位差がつくことはザラですが，12ビットのアナログ分解能は10μVといったオーダになることを忘れないでください．

(2)　基準電圧源の温度特性はどうか

図3.39の例ではICL7109内部の基準電圧源を使用していますが，これの温度係数は80 ppm/℃であり，0～50℃で使うとすれば0.4％もの誤差になります．これに対して外付けの基準電圧源を使えば1 ppm/℃くらいまでは得られます．これならば50℃の温度範囲に対して12ビットの確度を保てます．

このように，広い温度範囲で12ビット以上の確度を得るのは案外大変で，基準電圧源以外にも *CR* の温度特性などを抑え込まねばならない場合もあります（2重積分型では *CR* の温度特性は原則として誤差にはなりません）．

コラムF　デコーダのいろいろ

デコーダは入力が特定の条件になったことを検出すれば良い訳ですから，本質的には単なるAND（またはNAND）ゲートです．問題は，入力線数が非常に多いことです．たとえばZ80の場合，$\overline{\mathrm{RD}}$，$\overline{\mathrm{MREQ}}$，アドレス(の一部)を使用する必要があります．

たとえば図F.1の基本的なバス・インターフェースをアドレスF000H～F0FFHで使用するためのデコーダを考えてみます．図(a)はゲートを組み合わせたもの，(b)，(c)はアドレスをジャンパやDIPスイッチで設定できるものです．(c)のオープン・コレクタ型Ex-NORゲート（74LS266）はこのような一致検出に便利ですが，ワイヤードANDが可能なゲート数には制限があります．(d)はコンパレータを使用したもの，(e)はデコーダ用MSIを使ったもの，(f)はPAL（Programmable Array Logic）を使ったものです．コンパレータはTTL，CMOSともに種類が増えてきており，今後はもっと多く使われるでしょう．また，PALやPROMは，あらかじめ書き込んでおく必要がありますが，複雑なデコード回路でも1個のICで実現できます．

コラムG　フル・デコードとリニア・デコード

メモリやI/Oポートにアドレスを割り当てるとき，必要最小限のアドレス空間しか与えないようにアドレスをデコードすることを“フル・デコード”といいます．

たとえば，4KバイトのROMを64Kバイト（2^{16}）のアドレス空間に置く場合，フル・デコードするには少なくとも上位4本のアドレスをデコードする必要があります(図G.1：93ページ参照)．また，本文中の図3.6のような1バイトの入力ポートを，同じアドレス空間にフル・デコードで割り当てるには，16本のアドレスのすべてを用いてデコードを行う必要があります．

フル・デコードを行った場合の利点は，ムダなアドレスを生じないことです．アドレス空間に余裕が少ない場合や，今後の装置拡張の見通しがつかない場合，アドレス空間をなるべく有効に使用するためフル・デコードとします．

一方，アドレス空間に余裕があればフル・デコードする必要はありません(図G.2)．本文の図3.6の例では1バイトの入力ポートに256バイトの空間が割り当てられてい

図 F.1　デコーダのいろいろ

(a)　ゲートの使用例　(b) ジャンパの使用例　(c) EX-NORゲートの使用例

(d)　コンパレータの使用例　(e)　デコーダの使用例　(f)　PALの使用例

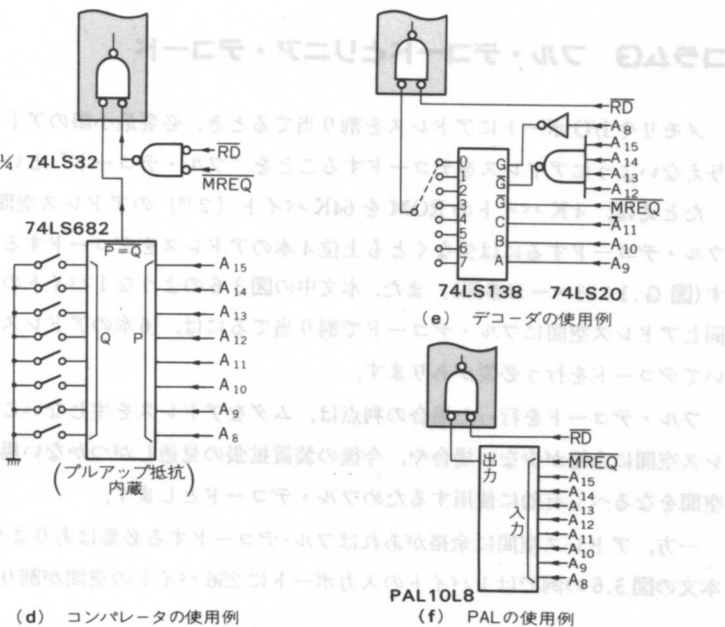
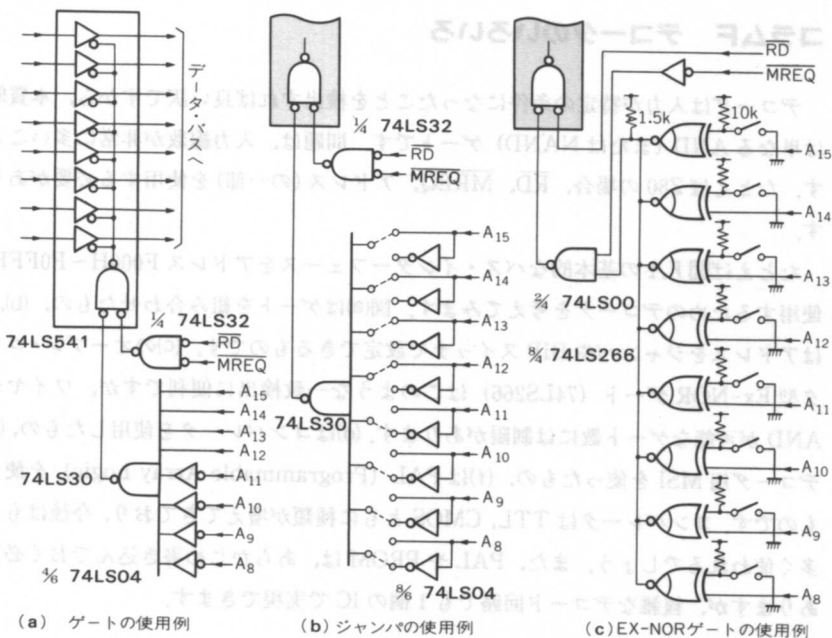



図 G.1　フル・デコードされたシステム

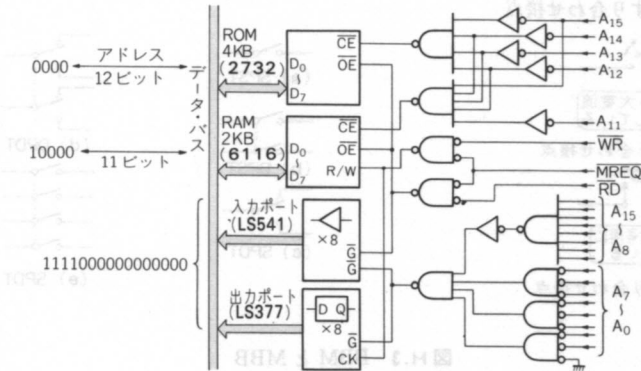
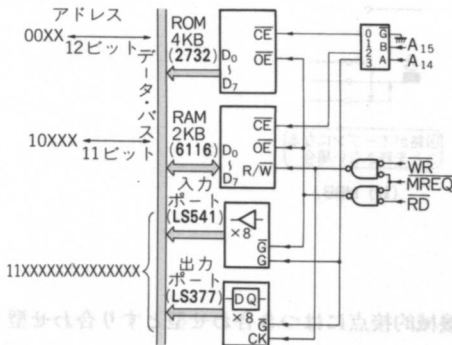
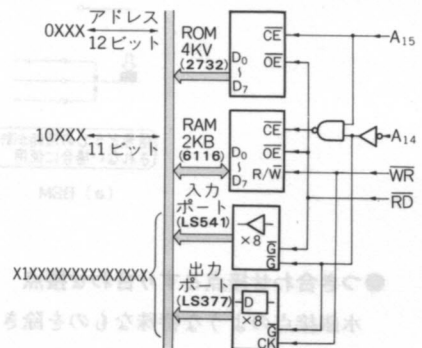



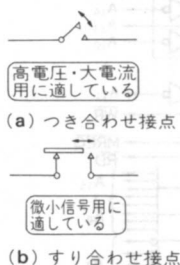
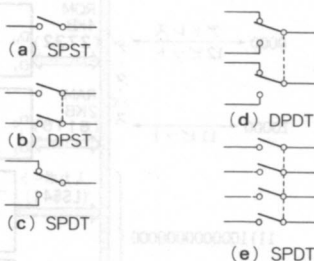
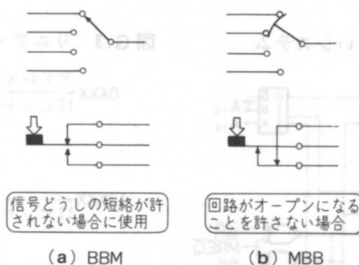
図 G.2　フル・デコードではないシステム　　　図 G.3　リニア・デコードを用いたシステム

ます．このほうがデコード回路が簡単になるからです．

この考え方のもっとも極端なものが“リニア・デコード”です．これはデコード回路をとくに設けず，アドレス線を各素子のイネーブル端子に直接に接続してしまおうというもので，ハードウェアはもっとも簡単になります．ただし，ソフトウェアのミスにより出力が衝突する危険もあるので注意が必要です（図 G.3）．

コラムH　スイッチの種類について

スイッチに関していろいろな分類の方法があります．ここでは，つぎの四つの分類法について示します．

図H.1　つき合わせ接点とすり合わせ接点
(a) つき合わせ接点
(b) すり合わせ接点

図H.2　SPST, DPDT, SPDT など
(a) SPST
(b) DPST
(c) SPDT
(d) DPDT
(e) SPDT

図H.3　BBM と MBB
(a) BBM
(b) MBB

●つき合わせ接点とすり合わせ接点

水銀接点のような特殊なものを除き，機械的接点にはつき合わせ型とすり合わせ型とがあります（図 H.1）．

つき合わせ型はマイクロ・スイッチやトグル・スイッチに多く用いられ，大電流・高電圧に耐えることを特徴とします．欠点は微小信号に対して接触不良を起こしやすいことです．

一方，すり合わせ型はスライド・スイッチやロータリ・スイッチに多く見られます．接点が接触したままある程度の距離を徐々に移動しますので，大電流や高電圧は使用できません．しかし．接点の自己清浄作用があるため，微小信号でも接触不良を生じにくいという長所があります．

● SPST，DPDT など

SPST は Single-Pole Single-Throw の略で「単極単投」と訳されます．DPDT は Double-Pole Double-Throw の略で「双極双投」と訳します．これらの意味は図 H.

2に示すとおりです．

● BBM，MBB

複数の接点を切り換える場合，隣の接点に接続する（make）前に，もとの接点との接続が切れる（break）ことがBreak before MakeでBBMと略されます．この逆がMake before Break（MBB）です．また，BBMはNon-Shorting，MBBはShortingともいいます（図H.3）．

● NOとNC

スイッチになにも力を加えない状態で開いているものをNormally-Open（略してNO），逆に閉じているものをNormally-Closed（NC）といいます．とくにNCは，Non-Conectionとまちがえやすいので注意．

第4章

マイコン出力インターフェースの設計

この章ではマイコンから外部機器にデータを出力する方法を説明します．

出力用インターフェースの一般形を図4.1に示します．ディジタル信号を出力するには，普通，データ・バス上の値をDフリップフロップなどで固定します．そのあと，必要に応じてデコードなどを行い，所定の形態およびレベルに変換します．

一方，アナログ信号は一般にバス上の値を固定したあとディジタル-アナログ変換器（D-A変換器）に印加されてアナログ出力となります．

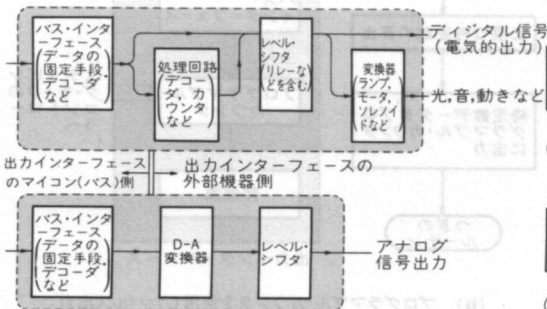
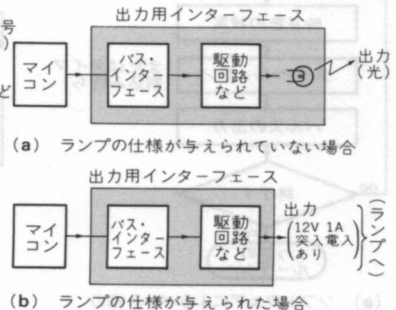
ただし，マイコンからの出力はデータ・バス上の値を固定して使うだけとはかぎりません．たとえば，データ・ストローブ信号を利用してパルス列を出力したり，アドレス信号を固定して並列データとして用いるなど，様々なバリエーションがあります．

また，発光ダイオード（LED）や液晶などの表示器の駆動回路も出力インターフェースに含めて考えるべきでしょう．

図4.1　出力用インターフェースの一般形

図4.2　出力用インターフェースの範囲

（a）ランプの仕様が与えられていない場合

（b）ランプの仕様が与えられた場合

本章では以上のようなことから，様々な出力インターフェースの事例を紹介します．

4.1 出力信号の種類と処理方法

マイコンから出力しようとする信号にも，入力信号に匹敵する種類があります．ここでは第1章に示した各項目の内容を簡単に説明しておきます．

●出力信号の形態とレベル

出力信号の形態やレベルの分類を行う場合，考えなければならないのはつぎの2点です．

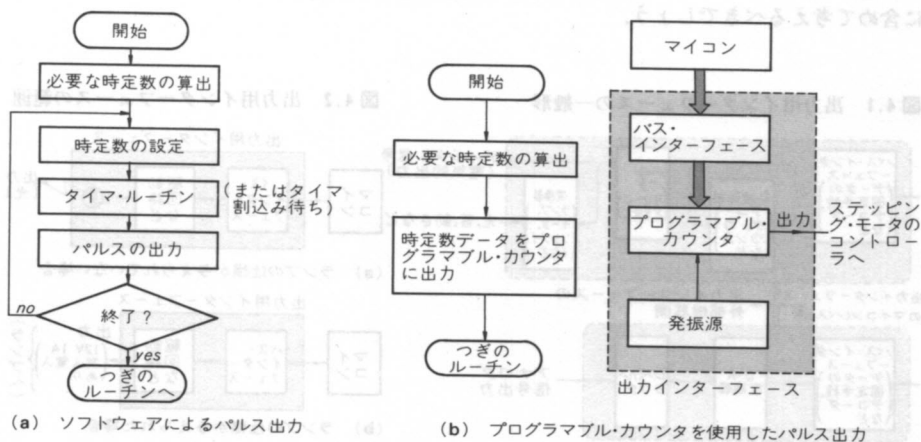
(1) 出力用インターフェースの範囲の考え方が一通りでない．

(2) 出力信号は，入力信号と異なり電力を考える必要がある．

まず(1)の問題については図4.2を用いて説明します．これは，ランプを点灯する例ですが，出力信号を“可視光線を利用した表示”と考えた場合，必要な明るささえ得られれば何V何Aの電球を使おうが，あるいはLEDを使おうが，インターフェース設計者の自由です．ところが，使用する電球の定格がすでに決定されている場合，インターフェース回路の扱う信号は［電流・電圧］で示されます．

本章ではモータやソレノイドなど機械的運動への変換器を割愛する点を除き，様々な出力信号形態を取り扱います．

図4.3 可変周波数のパルス出力の例

(a) ソフトウェアによるパルス出力　(b) プログラマブル・カウンタを使用したパルス出力

ただし，ここで注意しておきたい点は，出力インターフェースを設計する際には，たんに与えられた仕様にこだわらず，“最終的に必要な出力形態は何か？”をつねに考えるべきだという点です．

つぎに(2)の問題を考えます．外部からの信号をマイコンが取り込む際には，受け側の入力インピーダンスがある程度高ければ，電力的な条件を考える必要はありません．しかし，出力インターフェースは外部機器に対して電力を供給したり，電力を制御したりする必要があります．したがって，信号のレベルもつねに電力を含めて(たとえば電圧と電流とを)考えなければなりません．

●出力信号の速度

出力信号の速度として，まず第1に出力信号の状態を変化させる頻度や周期を考えることにします．

たとえば，ステッピング・モータ（パルス・モータ）のコントローラに対して印加するパルス信号をソフトウェアで作れるかどうかを考えてみます．図4.3(a)に示すのは1例で，出力パルス信号の間隔はループを1回まわる時間で制限されます．この場合，大体10 kHz（周期100 μs）程度が限界となります．これは入力信号を取り込める最短周期と同じオーダです．

では，これ以上のパルス周波数が必要な場合，どうすればよいのでしょうか．図4.3(b)にその1例を示しますが，これはプログラマブル・カウンタを使用してソフトウェアに無関係にパルス列を出力する回路です．こうすることによって，ソフトウェアの速度とは関係なく，高い周波数のパルスでも出力することができます．

ところが，この周波数を変化させようとすれば，やはりループをまわる時間が問題になります．したがって，ソフトウェアで判断や演算を行うかぎり，一定の時間より速く出力状態を変化させることはできません．これが，出力信号の状態を変化させる周期の限界であるといえます．

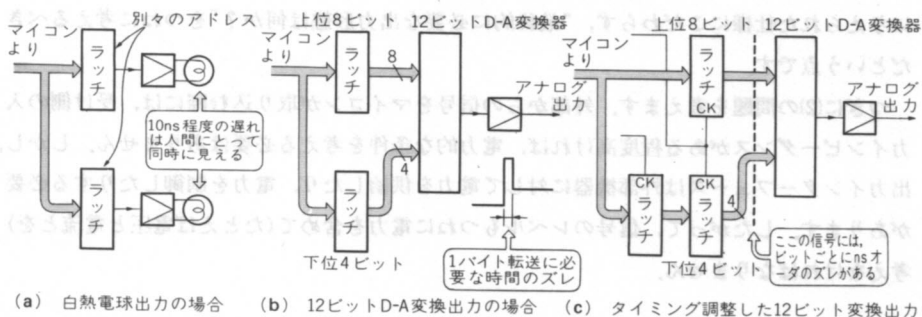
●出力信号の同時性

つぎに，出力信号の同時性について考えてみます．

複数の信号を“同時に”変化させなければならないことはよくあります．しかし，厳密な意味で“同時”というのは現実的ではありません．

たとえば図4.4(a)の白熱電球による表示の場合，人間の感覚の遅れや電球の時定数を考

図4.4　出力信号の同時性が要求される場合

(a)　白熱電球出力の場合　(b)　12ビットD-A変換出力の場合　(c)　タイミング調整した12ビット変換出力

えると，10msくらいの差で点灯してもとくに問題はありません．したがって，2個の電球に別々のアドレスが割り当てられていても，マイコンから“同時に”点灯することができます．

ところが，図4.4(b)の12ビットD-A変換器の場合，上位8ビットと下位4ビットとに別々のアドレスを割り当てておくと，アナログ出力を変更しようとするたびに大きなグリッチ（出力に生じるヒゲ）を生じ，使用目的によっては使いものになりません．

普通，このような場合には同図(c)に示すようにラッチを2段にするなどしてタイミングの不均一をなくします（例4-4を参照）．ところが，1個のICの中でも個々のビットの信号が同時に変化するという保証はありません．また，D-A変換器内部の速度のズレや配線などの影響により，各ビット間に最大で数10nsのタイミングのズレがあり得ます．この程度の時間が問題となる場合，アナログ信号をホールドするなど，さらに別の対策が必要です．このように“同時”といってもその程度によって様々の対策が必要です．

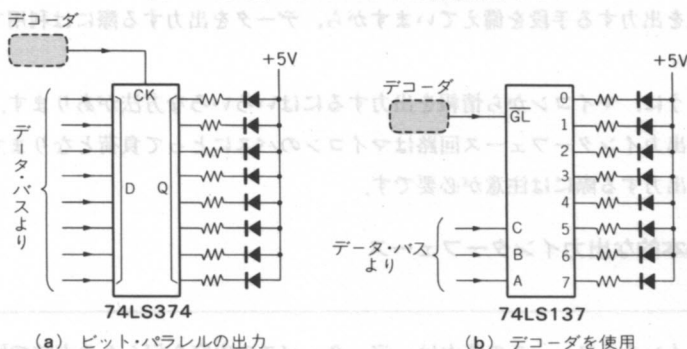
●出力信号の数

マイコンからの出力では，インターフェース回路からデータ・バスにデータを乗せることは(DMAを除けば)普通はありません．したがってバッファさえ設ければマイコンのバスに与える影響は問題ありません（例4-3を参照）．

しかし，きわめて多数の信号（たとえばマトリクス状のLEDの駆動信号）を出力する場合，信号線の本数や消費電力の削減のため，つぎのような事項が使えないかどうかを考えてみます．

(1)　デコーダや簡単な論理演算により，マイコンのバスに接続する信号線を減らすこと

図 4.5　多数の LED のうち 1 個を点灯する

(a) ビット・パラレルの出力　　(b) デコーダを使用

ができないか？

たとえば，多数の LED のうち 1 個だけを点灯するのであれば，図 4.5(a)よりも同図(b)の構成のほうが適しています．

(2) 表示器のようなものの場合，ダイナミック駆動を行うことで外部機器側の信号線を減らすことができないか？

とくに表示関係では各種のデコーダやダイナミック駆動用 IC が入手できるので，なるべくそれらを利用するよう心がけます（例 4-9，例 4-10 を参照）．

4.2　出力インターフェースのマイコン側の設計

外部に何らかの信号を出力するためのインターフェース回路の，マイコン（バス）側について考えてみます．この部分はつぎのような方法が考えられます．

(1) **データ・バスの値を固定する**　　もっとも一般的なのはこの方法です．マイコンのデータ・バスは時分割で様々な情報が乗っているので，必要とする時点の値を D フリップフロップなどで固定します．このために，アドレスや制御信号の値をデコードする回路が必要です．

(2) **制御信号を使う**　　パルス状の出力信号が必要な場合，アドレスや制御信号をデコードしたものを，そのまま出力として使うことができます．

この場合，データ・バスの内容は何でもかまわないので，書込み（$\overline{\mathrm{WR}}$）信号にかぎらず読出し（$\overline{\mathrm{RD}}$）も利用可能です．

(3) **アドレス・バスの値を固定する**　　CPU のアドレス・バスは出力専用のバスと見るこ

とができます．多くのCPUではレジスタ間接アドレシングを始めとして，アドレス・バスに任意の値を出力する手段を備えていますから，データを出力する際には利用できます．

以上のように，マイコンから情報を出力するにはいろいろな方法があります．いずれの場合にも，出力インターフェース回路はマイコンのバスにとって負荷となりますから，多数の信号を出力する際には注意が必要です．

例 4-1　基本的な出力インターフェース

出力用インターフェースの基本は，データ・バスの内容を何らかの方法で固定することです．

ここでは，データ・バスと内容を固定する基本的な方法と，タイミングの計算方法とを説明します．

図4.6に示すのは，基本的な出力インターフェースの例です．

▶デコードとデータ・バスの内容の固定　一般に出力インターフェースのマイコン側にはつぎのような要素が必要です．

(1)　アドレスや制御信号が所定の組合せになったことを検出するデコーダ．

図4.6　基本的出力インターフェース

(2)　(パルス状の出力信号でよい場合を除いて) データ・バスの内容を固定する手段.

まず，デコーダについては例3-1を参照してください．普通は読出し信号 ($\overline{RD}$) のかわりに書込み信号 ($\overline{WR}$) を使います (あたりまえ)．しかし，出力用インターフェースはデータ・バスに信号を乗せるものではありませんから，特殊な使い方として $\overline{RD}$ を用いることもできます．

前記(2)のデータ・バスの内容を固定する手段ですが，普通はDフリップフロップ (コラム1参照) またはトランスパレント (透過型)・ラッチ (コラム1参照) を用います．ただし,どのタイミングでデータを固定しなければならないかはCPUによって，あるいは使い方によって異なりますので別途に検討しなければなりません (後述)．

▶CPUに接続するための条件　つぎに，図4.6(b)の回路を実際にCPUに接続できるかどうかを調べてみます．この場合に考慮しなければならないのはつぎの2点です．

(1)　この回路を接続することによってCPUや他のバス・スレーブに与える影響.

(2)　この回路がマイコンのバス上の信号を固定できるか否か.

▶バス・スレーブに与える影響　まず前者ですが，TTLを使用するかぎり，出力用インターフェースのほうが入力用よりもデータ・バスに大きな影響を与えます．これは，おもにTTLの入力電流に起因します．

たとえば図4.6(a)の74LS377の場合，入力電流は，

"H" 側：$I_{IH}=20\mu A$(max) (流込み)

"L" 側：$I_{IL}=-400\mu A$ (max) (流出し)

です．一方，Z80A CPUのデータ・バスの駆動能力は，

"H" 側：$I_{OH}=-250\mu A$ ($V_{IH}=2.4V$)

"L" 側：$I_{OL}=1.8mA$ ($V_{OL}=0.4V$)

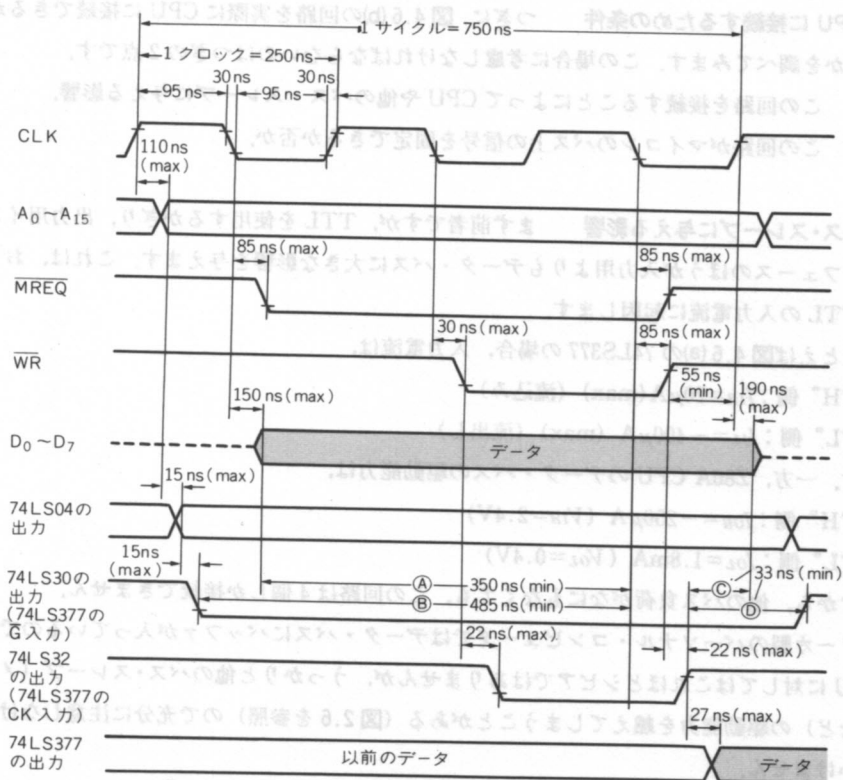
ですから，他のバス負荷がなにもなくても，この回路は4個しか接続できません．

メーカ製のパーソナル・コンピュータではデータ・バスにバッファが入っているので，CPUに対してはこれほどシビアではありませんが，うっかりと他のバス・スレーブ (メモリなど) の駆動能力を越えてしまうことがある (図2.6を参照) ので充分に注意しなければいけません．

▶バス上の信号の固定法　つぎに，この回路がマイコン・バス上の信号を正しく固定で

きるか否かを調べてみます．

図4.7に示すのはZ80A CPUのメモリ領域に図4.6(a)の回路を接続した際の具体的なタイミングです．Z80のメモリ領域ではデータが出力されてから$\overline{\mathrm{WR}}$の前縁（立下り）まで1クロック，後縁（立上り）までならば2クロックの時間があります．各バスにバッファが挿入されているか否か，あるいはデコード回路の形によって多少の変化はありますが，Z80A CPU(4 MHz)のメモリ領域で$\overline{\mathrm{WR}}$信号の後縁を使えば，大体300ns以上のデータ・セットアップ時間が確保できます（図4.7の場合はⒶの350ns）．また，データ・ホールド時間もかなり得られます(Ⓒ 33ns)．これはTTLのDフリップフロップでデータを固定す

図4.7　Z80Aを用いたときのデータ出力タイミング

るには充分な値ですし，トランスパレント・ラッチも使用可能です．

一方，同じZ80AでもI/O領域では$\overline{WR}$の前縁に対してはデータ・セットアップ時間が確保できません．したがって，図4.6(b)の回路はI/O領域には使用できません．*

例4-2 ビット単位でオン/オフできる出力インターフェース

他の出力信号の状態を変化させることなく，1本の出力信号だけを変化させたいことがあります．

このような場合，データ・バス中の1ビットだけを使用して各信号を別々のアドレスに振り分けるのが簡単です．

リレーやランプを駆動する場合，1個ずつ独立にオン/オフできたほうがよいこともあります．そのような場合，図4.8に示すようにアドレサブル（ビット指定可能）・ラッチを使用する方法があります．

アドレサブル・ラッチは図4.9のようにデコーダと各ビットのラッチとが組み合わされ

図4.8 ビット単位でオン/オフできる出力インターフェース

（a）回路例

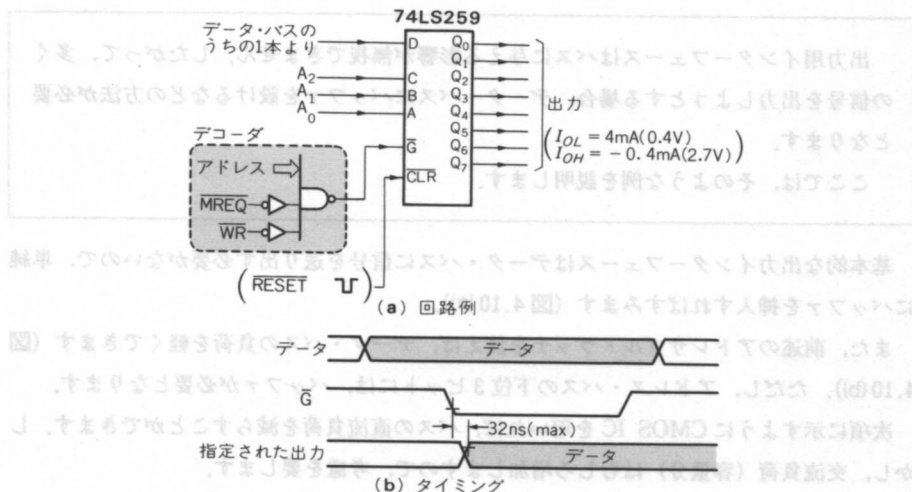
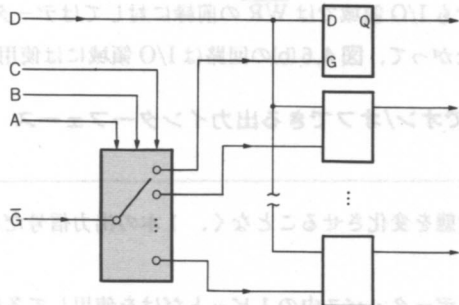
（b）タイミング

* クロック周波数やそのデューティ比を選べば，そのかぎりではない．

図4.9 アドレサブル・ラッチの内部構成

たものです．図4.8(a)の74LS259はTTLでは代表的なアドレサブル・ラッチで，$\overline{G}$が"L"の時点ではDに印加したデータが出力（Q_0～Q_7）のうちの1本（3ビットのアドレスで指定する）につつ抜けになります．つまり，ラッチ機能としてはトランスパレント・ラッチですので，出力不確定の時間が生じないように注意します．たとえば，4MHzのZ80A CPUのI/O領域には，このまま直結すると出力不確定の時間が生じ得ます．

例4-3 多数の信号を出力するインターフェース

出力用インターフェースはバスに与える影響が無視できません．したがって，多くの信号を出力しようとする場合，データ・バスにバッファを設けるなどの方法が必要となります．

ここでは，そのような例を説明します．

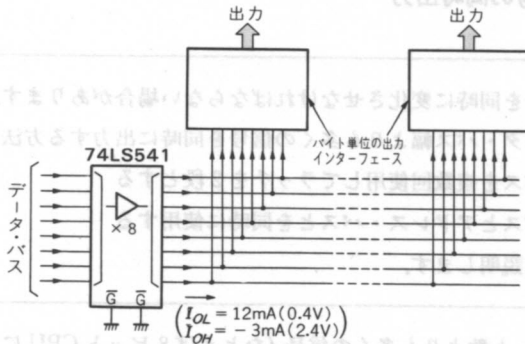
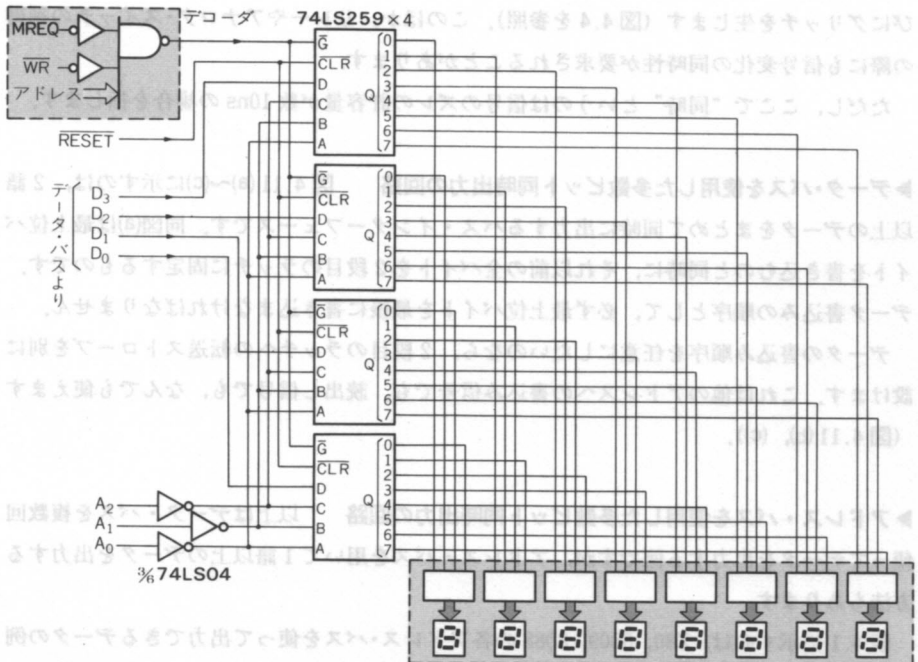
基本的な出力インターフェースはデータ・バスに信号を送り出す必要がないので，単純にバッファを挿入すればすみます（図4.10(a)）．

また，前述のアドレサブル・ラッチを使えば，データ・バスの負荷を軽くできます（図4.10(b)）．ただし，アドレス・バスの下位3ビットには，バッファが必要となります．

次項に示すようにCMOS ICを用いれば，バスの直流負荷を減らすことができます．しかし，交流負荷（容量分）はむしろ増加しますので，考慮を要します．

図 4.10 多数の信号を出力する

(a) バッファを挿入しただけの例

(b) アドレサブル・ラッチを使用した例　　7セグメント・デコーダおよびLED駆動回路(8桁分)

例 4-4　多数の信号の同時出力

多数の出力信号を同時に変化させなければならない場合があります．
ここでは，データ・バス幅よりも多くの信号を同時に出力する方法として，
(1)　データ・バスを複数回使用してラッチを2段とする
(2)　データ・バスとアドレス・バスとを同時に使用する
の2通りについて説明します．

データ・バスのビット数よりも多くの信号（たとえば8ビットCPUに対して9本以上の信号）を同時に変化させなければならない場合もあります．

代表的なのはD-A変換器で，すべてのビットを同時に変化させないと，データ変更のたびにグリッチを生じます（図4.4を参照）．このほか，リレーやアナログ・スイッチの制御の際にも信号変化の同時性が要求されることがあります．

ただし，ここで“同時”というのは信号のズレの許容量が数10nsの場合を指します．

▶データ・バスを使用した多数ビット同時出力の回路　図4.11(a)～(c)に示すのは，2語以上のデータをまとめて同時に出力するバス・インターフェースです．同図(a)は最上位バイトを書き込むのと同時に，それ以前の全バイトを2段目のラッチに固定するものです．データ書込みの順序として，必ず最上位バイトを最後に書き込まなければなりません．

データの書込み順序を任意にしたいのなら，2段目のラッチへの転送ストローブを別に設けます．これは他のアドレスへの書込み信号でも，読出し信号でも，なんでも使えます（図4.11(b)，(c)）．

▶アドレス・バスを使用した多数ビット同時出力の回路　以上はデータ・バスを複数回使ってデータを出力する例ですが，アドレス・バスを用いて1語以上のデータを出力する方法もあります．

表4.1に示すのは，Z80，6809，8088の各アドレス・バスを使って出力できるデータの例です．いずれのCPUもレジスタを使った演算の結果をアドレス・バスに出力することができますから，見方によっては24ビット（Z80，6809）または28ビット（8088）までの信号は同時に出力可能です．

図 4.11　多数ビットの同時出力

(a)　データの書込みを下位バイトから行う例

(b)　データの書込み順序を任意に行う例①

(c)　データの書込み順序を任意に行う例②

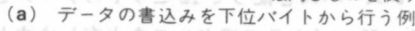
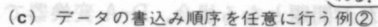



表4.1　アドレス・バスの内容を出力として使用する

CPU	命令(例)	アドレス・バス	データ・バス
Z80	LD (nn), A	M　イミディエイト	A
	LD (HL), A	M　HL	A
	LD (BC), A	M　BC	A
	OUT(BC), A	I/O　BC	A
6809	STA (ダイレクト)	DP＋イミディエイト	A
	STA (Bオフセット)	X＋B	A
8088	MOV メモリ, A	DS BX＋DI	AL

図4.12(a)はZ80のアドレス・バスを使って12ビットのデータを一度に出力するものです．この場合，少なくとも4Kバイト分のアドレスを占有しますから，アドレス空間に余裕がないと使えません．

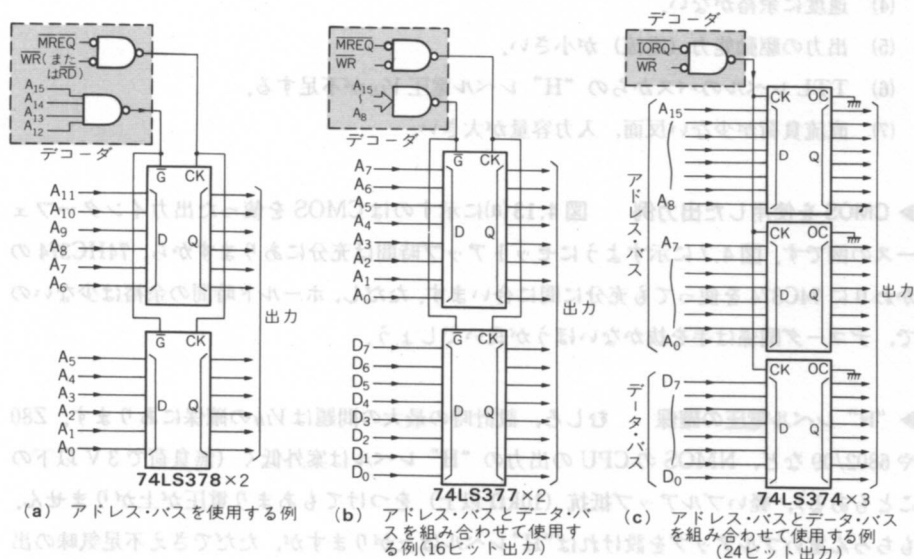
▶アドレス・バスとデータ・バスを組み合わせて使用した例　図4.12(b)はアドレス・バスとデータ・バスとの組合せによるものです．計16ビットの信号ならば占有アドレスは256バイト，計12ビットならば16バイトに過ぎません．

図4.12(c)はZ80のアドレス・バスとデータ・バスとの全幅を用いた24ビットの出力回路です．全幅を使ってしまっては困るのではないかと思われるかもしれませんが，RAMや他の出力ポートをすべてメモリ領域に置き，入出力領域にこの回路を配置すれば可能です（もちろん逆の配置も可）．

このように，アドレスに余裕さえあれば，8ビットCPUであってもそれ以上のビット数の同時出力が可能です．

なお，図4.11，図4.12のいずれにおいても，Dフリップフロップ(またはラッチ)の各ビットの遅れ時間の差は出力タイミングのバラツキとして必ず残ります．LSタイプのTTLの場合，標準的には数ns，74LS377の最大値は27ns（V_{CC}＝5V，T_a＝25℃）です．これが問題となる場合，D-A変換器であればアナログ信号をホールドする方法があります．

図4.12 アドレス・バスを使って多数ビットを出力する方法

（a） アドレス・バスを使用する例　（b） アドレス・バスとデータ・バスを組み合わせて使用する例(16ビット出力)　（c） アドレス・バスとデータ・バスを組み合わせて使用する例(24ビット出力)

例4-5 CMOS出力インターフェース

> CMOS ICには入力インピーダンスが高く，消費電流が少ないなどの利点があり，CMOSによるインターフェース回路も有用です．
>
> しかし，入力容量の増加やタイミング，論理レベルなど，考慮しなければならない問題が多くあります．
>
> 本項では，これらの問題について説明します．

例4-1に示したように，CPUから出力される各信号はタイミング的に余裕があります．したがって，TTLに比べて速度の遅いCMOSでも使用できる可能性があります．

出力インターフェースにCMOSを使った場合の利点としては，つぎのようなことが考えられます．

(1) バスに対して直流的な負担が少ない．

(2) 消費電流が少なくてすむ．

(3) 出力電圧が電源電圧いっぱいに振れる．

一方，CMOSがTTLに比べて不利な点はつぎのとおりです．

(4)　速度に余裕がない．

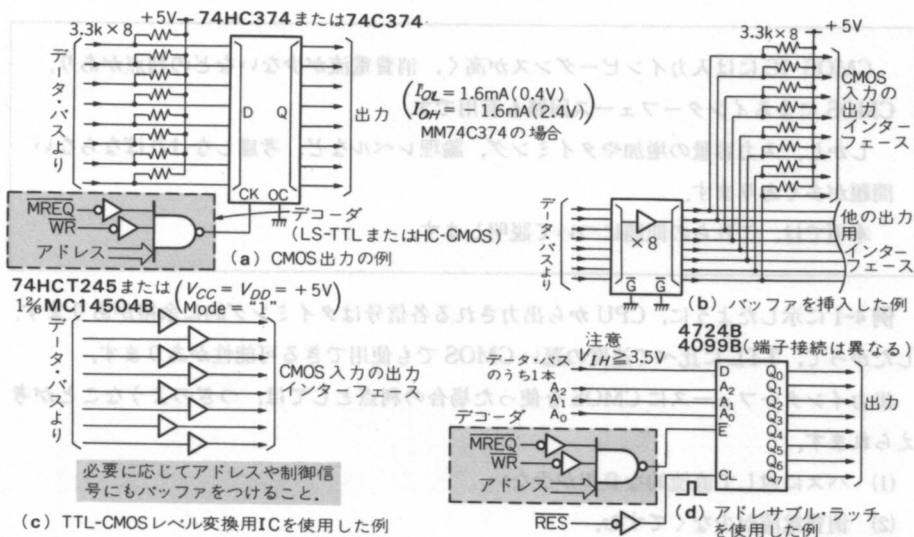
(5)　出力の駆動能力（電流）が小さい．

(6)　TTLレベルのバスからの“H”レベル電圧V_{IH}が不足する．

(7)　直流負荷が少ない反面，入力容量が大きい．

▶ CMOSを使用した出力例　図4.13(a)に示すのはCMOSを使った出力インターフェースの例です．図4.7に示すようにセットアップ時間は充分にありますから，74HC374のかわりに74C374を使っても充分に間に合います．ただし，ホールド時間の余裕は少ないので，デコーダ関係は手を抜かないほうが良いでしょう．

▶ “H”レベル電圧の確保　むしろ，設計時の最大の問題はV_{IH}の確保にあります．Z80や6802/09など，NMOSのCPUの出力の“H”レベルは案外低く（無負荷で3V以下のこともある），軽いプルアップ抵抗（10kΩ以上）をつけてもあまり電圧が上がりません．もちろん重いプルアップを設ければ“H”レベルは上がりますが，ただでさえ不足気味の出

図4.13　CMOSを使った出力用インターフェース

力駆動能力をムダ使いすることになります．

TTLのバッファが付いている場合（図4.13(b)）は，プルアップ抵抗で“H”レベルを引き上げることがむしろ容易です．なお，TTLからCMOSへの“H”レベル確保のためこのようにプルアップ抵抗を付けることがよく行われ，多くの場合，とくに問題なく動作します．しかし，TTLの出力をプルアップしたからといって V_{OH} (min) が高くなることが保証されているわけではありませんから念のため．

もっとも確実な方法は，MC14504Bや74HCTシリーズのようにTTL→CMOSのレベル変換用ICを挿入してしまうことでしょう（図4.12(c)）．

図4.13(d)に示すのは例4-2に相当するアドレサブル・ラッチです．4724Bは74LS259と同じ端子接続としたものです．ただし，リセットの極性が逆（アクティブ“H”）になっていますので注意しなくてはいけません．

例4-6　パルス信号の出力

外部機器のなかには，ステッピング・モータのコントローラのように，パルス状の信号を必要とするものがあります．これ以外にも，ハンドシェイク用のストローブやアクノリッジ信号，フリップフロップのリセット信号，警報音を出すためのスピーカへの信号，電磁カウンタへの信号など，パルスの個数または周波数が問題となる用途は少なくありません．

このような場合，一定のレベルの信号を出力するのとは異なった方法も使えます．ここでは，それらについて説明します．

例4-2で述べた回路を用いてソフトウェアで“1”と“0”とを交互に出力すれば，確かにパルス信号が出力できます．セントロニクス型出力のストローブ信号や，スピーカからブザー状の音を出す場合など，簡単にはこの方法が使えます．

しかし，この方法ではCPUがパルス間隔やパルス数の面倒を全部みてやる必要があり，周波数の上限も数10 kHzどまりでしょう．

▶データ・ストローブを直接使用する方法　図4.14(a)に示すのはCPUのデータ・ストローブを直接に用いるものです．データ・ストローブ信号としては，アドレスをデコードしただけというのは感心しません（アドレス各ビットの遅れ時間のズレがあるため）．パルス

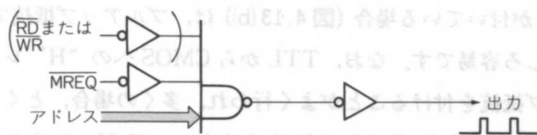
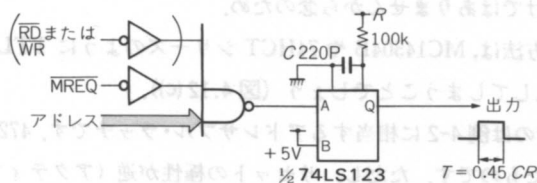
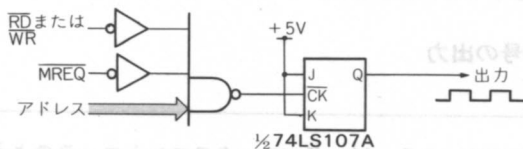
図4.14　パルス出力用インターフェース

(a)データ・ストローブを直接使用した例

(b)ワンショットを挿入しパルス幅を長くする例

(c) 1回ごとに出力を反転する例

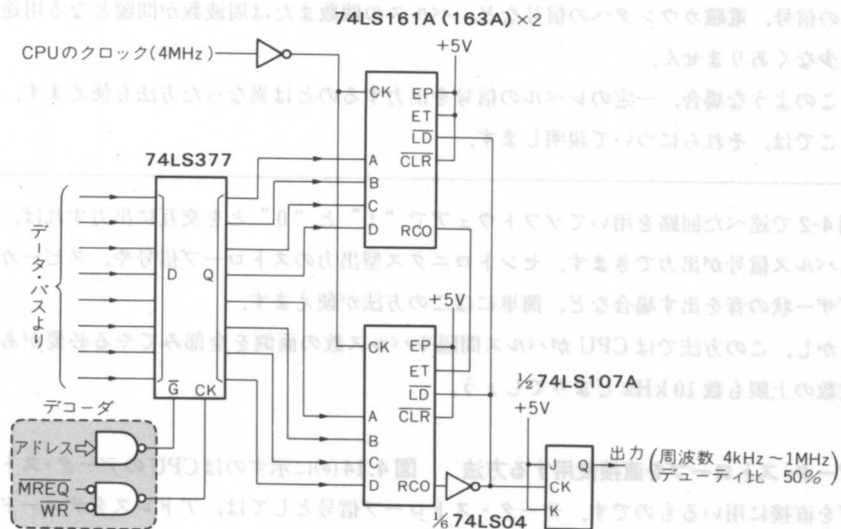
図4.15　プログラマブル・カウンタを用いた出力インターフェース

幅は$\overline{RD}$または$\overline{WR}$信号で決まり，4 MHzのZ80Aのメモリ領域では375ns (min) となります．

図4.14(b)はパルス幅を延長するのにワンショットを付加したもの，同図(c)は1回ごとに出力が反転するものでデューティ比50％のパルスが必要な場合（スピーカを鳴らすには，パワー的に50％が有利）に使います．ただし，これらの回路は雑音に弱い欠点があることを承知しておく必要があります．

▶プログラマブル・カウンタを使用する方法　以上の回路では出力パルスの周期は，やはりCPUのループ周期で決められてしまいます．それに対して図4.15の回路はCPUから分周比を定めるものです．8253やZ80 CTCは基本的にはこのような回路となっています．直列通信回線のボーレート用クロック発生や，簡単なスピーカ駆動，あるいはステッピング・モータの速度制御用など，用途は広いでしょう．

例4-7 ストローブ付き出力とセントロニクス型ハンドシェイク出力

> プリンタなどのインターフェースは2線または3線式ハンドシェイクとなっています．
>
> 本項では，基本となるストローブ付き出力と，セントロニクス型ハンドシェイク出力とについて，実際的な構成法を紹介します．

マイコンのバス自体がストローブ付き出力の形になっていますから，ストローブ・パル

図4.16　ストローブ付き出力

(a) 回路例　(b) タイミング

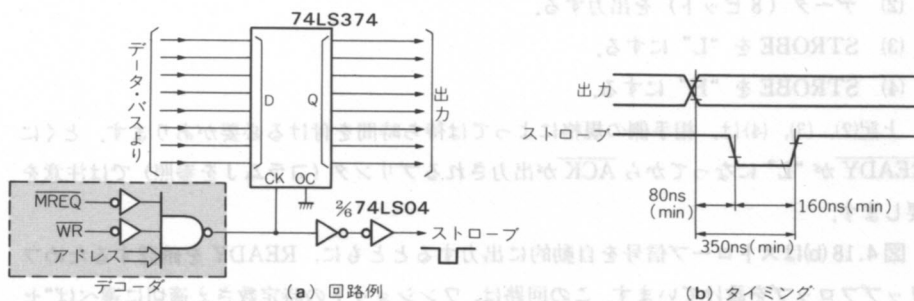
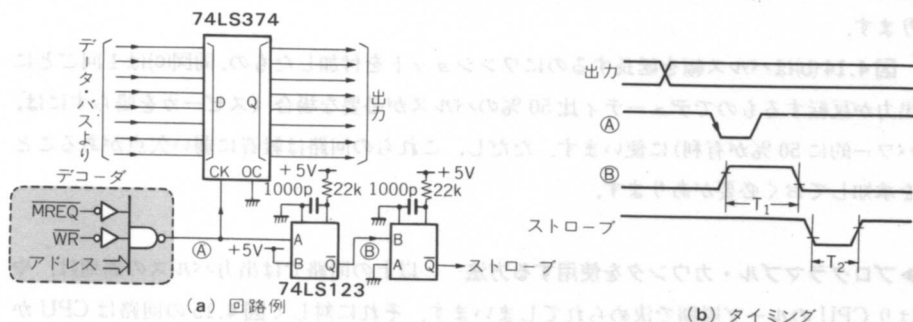



図 4.17　タイミングを延長できるストローブ付き出力

(a) 回路例

(b) タイミング

ス幅やセットアップ時間，ホールド時間が要求を満足するのであれば，図4.16の回路がそのまま使えます．また，図4.17のようにすればタイミングを延長できます．

▶セントロニクス型インターフェース　一方，いわゆるセントロニクス型インターフェース（2線または3線式ハンドシェイク）は，プリンタやプロッタの多くが採用しており，マイコンからの出力としてよく使用されます．ただし，セントロニクス型といっても実際の個々の仕様は変化に富んでおり(コラムJ参照)，すべてに適用できる出力回路は作り得ません．したがって，相手側の規格をよく調べて，それに合致するものを設計するようにします．

図4.18(a)はもっとも簡単なもので，市販のメーカ製パーソナル・コンピュータの多くもこの程度の回路となっているようです．この回路では，データを出力するごとにつぎのようなプログラムが必要となります．

(1)　BUSY/$\overline{\mathrm{READY}}$が“L”かどうかを調べ，“H”ならば“L”になるのを待つ．

(2)　データ（8ビット）を出力する．

(3)　STROBEを“L”にする．

(4)　STROBEを“H”にする．

上記(2)，(3)，(4)は，相手側の規格によっては待ち時間を付ける必要があります．とくに$\overline{\mathrm{READY}}$が“L”になってから$\overline{\mathrm{ACK}}$が出力されるプリンタ（コラムJを参照）では注意を要します．

図4.18(b)はストローブ信号を自動的に出力するとともに，READYを確認するためフリップフロップを設けています．この回路は，ワンショットの時定数さえ適切に選べば“セ

ントロニクス型"の大半をカバーできます．データの出力手順はつぎのとおりです．

(1) RDY 信号が "H" であるかどうかを調べ，"L" ならば "H" になるまで待つ．

(2) データを出力する．

なお，この手の TTL レベルの並列のインターフェースを数 m も引っ張るには，受け側で確実に終端（またはプルアップ）する必要があります．プリンタなどのなかには，この処置が行われていないものがありますし，逆に市販パソコンの側の駆動能力が充分でない

図 4.18 セントロニクス型ハンドシェイク出力

(a) 基本的なセントロニクス型出力回路

(b) 自動ストローブ信号出力，READY 確認つきセントロニクス型出力回路

(c) (b)のタイミング

ものもあるので注意したいものです．8255 や 6821, Z80 PIO などで直接に数 m のケーブルを駆動するのは無理があります．

4.3 出力インターフェースの外部機器側の設計

出力インターフェースのマイコン側ポートによって，バスから必要な信号を取り出します．この出力は，TTL または CMOS レベルで数 mA の駆動能力をもっています．しかし，出力しようとする信号には多くの種類があり，様々な回路を付加することが必要となります．このような外部機器各ポートは，つぎのような役目をもっています．

(1) **信号の形態およびレベルの変換**　TTL レベルや CMOS レベルの信号を，接点やオープン・コレクタ，あるいは大電力のスイッチ回路など，様々な形態に変換する．また，アナログ出力が必要な場合には D-A 変換器を接続する．

(2) **信号の多重化などを行う**　発光ダイオード（LED）や液晶表示器（LCD）には，ダイナミック駆動を用いたほうが信号線の本数が少なくてすむ．また，並列データをビット・シリアル（直列）に直したほうが長距離の伝送に好都合なこともある．このような操作も出力インターフェースで行う必要がある．

本章では，以上のような点を念頭に置いて，出力インターフェースの外部機器側の例を以下に紹介することにします．

例 4-8　ランプ（白熱電球）の点灯

白熱電球は表示器として優れた特性をもつ反面，突入電流が存在するとともに，高電圧や大電流を取り扱わざるを得ないことも少なくありません．

リレーや SSR（ソリッド・ステート・リレー）を用いずに電球を駆動する場合の手法について，本項で説明します．

LED や液晶などの表示素子も急速に進歩していますが，見やすさの点では白熱電球を越えていません．そのため，短寿命・発熱・低効率・大型などの欠点にかかわらず，未だに電球も随所で使われています．

白熱電球を点灯するには電流を流しさえすれば良いのですが，電球のフィラメントは低温（消灯）時と高温（点灯）時とで，その抵抗が 1 桁以上も変化します．したがって点灯

の瞬間に定格の数十倍の突入電流が流れることになり，その対策がポイントとなります．

図4.19に示すのは12V 50mA程度の電球の点灯回路です．突入電流対策の基本は，

(1)　消灯時にも若干の電流を流しておく

(2)　点灯時には直列に抵抗を挿入して電流を制限する

のいずれか，または両方を行うことです．(1)の消灯時の電流はフィラメントがかすかに赤熱する程度，定格電流の10％のオーダを限度としますが，それでも突入電流を数分の1から1/10程度にまで減らせます．

(2)については効率の悪化を無視すれば，定電流駆動としてしまえば理想的です．

図4.20は12V 1A程度の駆動回路です．電流が増加すればダーリントンにしたいところですが，飽和電圧が高くなって損失が増し，効率は低下します．パワーMOS FET(図4.19(c)，図4.20(b))はディジタル回路との相性が非常に良く，また突入電流やサージ電圧に

図4.19　白熱電球の駆動――その1

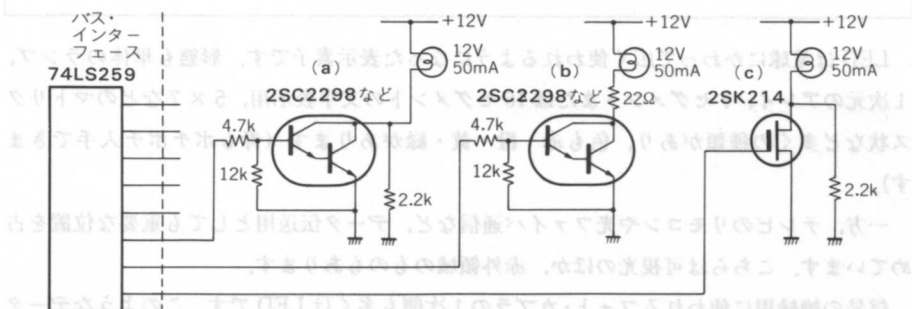



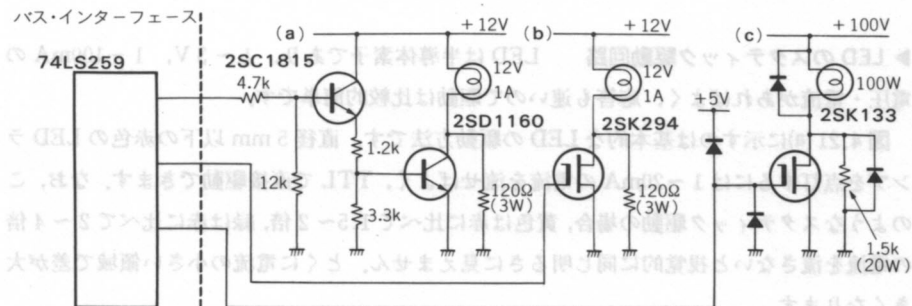
図4.20　白熱電球の駆動――その2

強いので，もっと使われてもいいと思います．

図4.20(c)に示すのは100V 1～2Aの駆動回路です．このクラスになるとリレーやSSR（ソリッド・ステート・リレー）を介することが多くなりますが，このように直接の駆動も不可能ではありません．ただし，マイコン側に高圧が絶対にまわり込まないよう，駆動用トランジスタの耐圧や絶縁，結線方法などに充分に注意する必要があります．

例 4-9 LEDの駆動

LED(発光ダイオード)は高輝度化，多色化，大型化が進み，電球にかわって表示用として広く用いられるようになりました．一方，データ伝送用としても高速・高効率化が進んでいます．

もともと，LEDはTTLなどのディジタルICと接続が容易なのですが，多桁表示やデータ伝送用の高速駆動にはそれなりの注意が必要となります．

LEDは電球にかわって広く使われるようになった表示素子です．形態も単体のランプ，1次元のアレイ，7セグメントまたは16セグメントの文字表示用，5×7などのマトリクス状など多くの種類があり，色も赤・橙・黄・緑があります（青もポチポチ入手できます）．

一方，テレビのリモコンや光ファイバ通信など，データ伝送用としても重要な位置を占めています．こちらは可視光のほか，赤外領域のものもあります．

信号の絶縁用に使われるフォト・カプラの1次側も多くはLEDです．このようなデータ伝送用のLEDはオン/オフの速度や発光効率が問われることが多く，人間に見えればよいという表示用とは多少異なります．

▶ LEDのスタティック駆動回路　LEDは半導体素子であり，1～3V，1～100mAの電圧・電流があればよく，応答も速いので駆動は比較的簡単です．

図4.21(a)に示すのは基本的なLEDの駆動方法です．直径5mm以下の赤色のLEDランプを点灯するには1～20mAの電流を流せばよく，TTLで直接駆動できます．なお，このようなスタティック駆動の場合，黄色は赤に比べて1.5～2倍，緑は赤に比べて2～4倍の電流を流さないと視覚的に同じ明るさに見えません．とくに電流の小さい領域で差が大きくなります．

7セグメントの数字表示器には専用のデコーダ用ICがあります．図4.21(b)はTTLのデコーダ/ドライバを使用した例です．字体と駆動能力とによって多くの種類があります．

一方，同図(c)はCMOSのデコーダ/ドライバを使用した例です．ラッチ付きのものもあ

図4.21 LEDのスタティック駆動

(a) 基本的な駆動回路 (b) デコーダ/ドライバを使用した駆動回路

(c) CMOSのデコーダ/ドライバを使用した回路

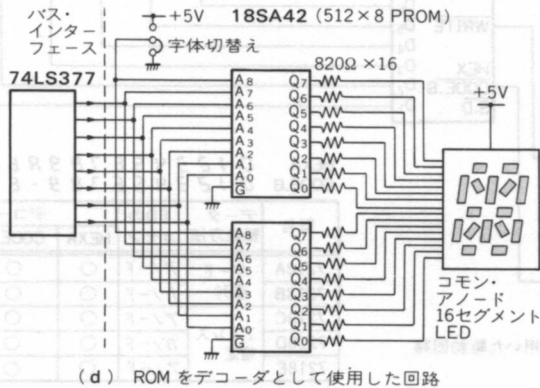
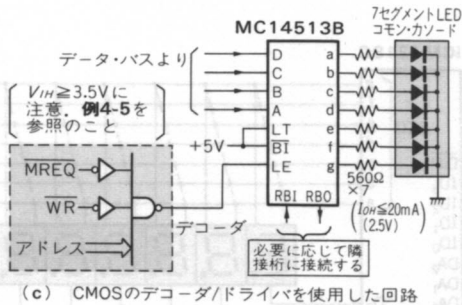
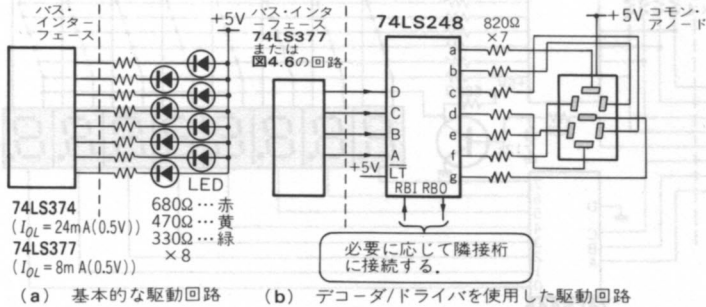
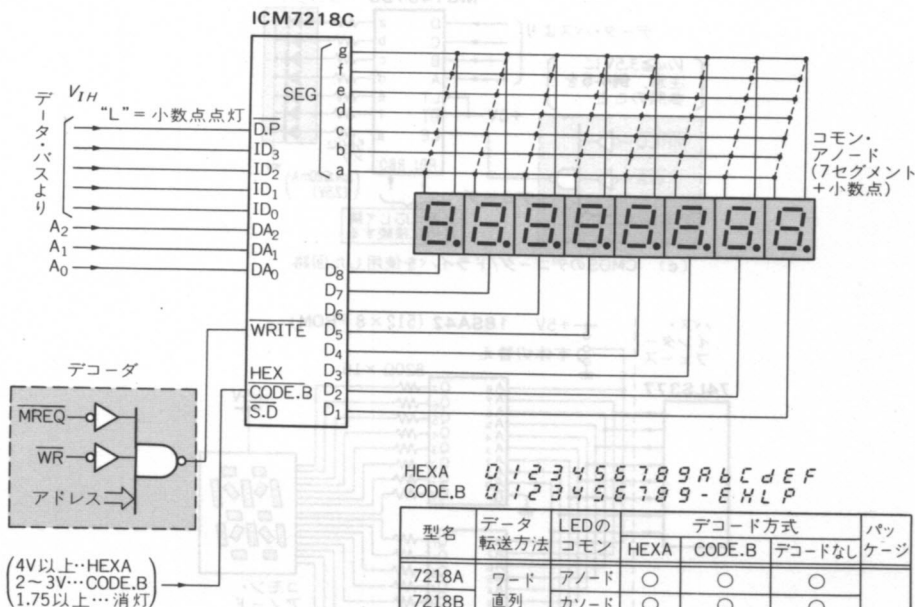
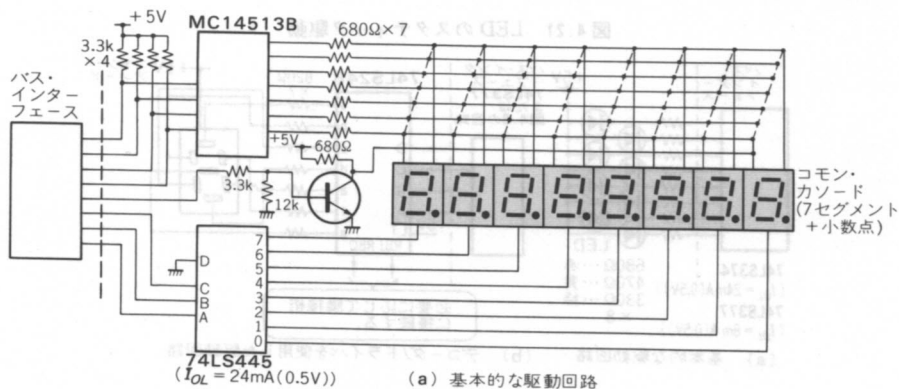
(d) ROMをデコーダとして使用した回路

りますので，CPUのバスに直結も可能です．CMOSで駆動能力が不足する場合はTTLやトランジスタで補強します．

図 4.22　LEDのダイナミック駆動

型名	データ転送方法	LEDのコモン	デコード方式			パッケージ
			HEXA	CODE.B	デコードなし	
7218A	ワード直列	アノード	○	○	○	28
7218B		カソード	○	○	○	
7218C	アドレス指定	アノード	○	○	—	
7218D		カソード	○	○	—	
7218E		アノード	○	○	○	40

16セグメントのものは数字および英文字が表示できます．しかし，7セグメント用のような簡便なデコーダICが入手できませんので，図4.21(d)のようにROMを使ってデコーダを作ると良いでしょう．もちろん，ソフトウェアでデコードしてもかまいません．

▶ LEDのダイナミック駆動回路　図4.22に示すのはダイナミック駆動の方法です．一般に，LEDも個数が多くなるとバス・インターフェースも配線も大変です．したがって，人間の眼の残光特性を利用したダイナミック駆動によって駆動信号の本数を減らします．

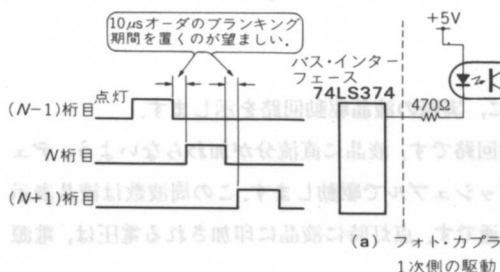
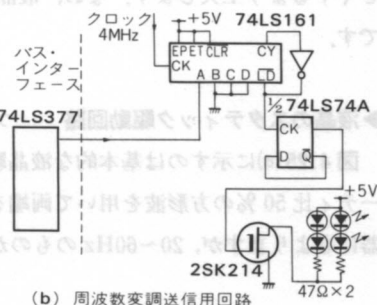
図4.22(a)に示すのは基本的なダイナミック駆動の回路です．これはソフトウェアによって一定時間ごとにCPUから値を書き込む必要があります．この周期は20ms以内にしないと，点滅をはっきり感じます．また，スキャンが速すぎると回路やLEDの遅れによって隣接桁が点灯します．本当は表示桁を切り換える際にブランキング期間を置くべきです（図4.23）．

図4.22(b)はダイナミック駆動の専用ICを用いた例です．このICでは64個までの単体のLEDランプ，または（7セグメント＋小数点）×8桁の数字表示器を駆動できます．データの転送方法，LEDのコモンがアノードかカソードか，デコードするしないかにより，7218A～Eまでの5種があります．CPUからはたんなる出力ポートに見えますが，内部メモリの読出しはできません．

▶データ伝送用LEDの駆動回路　つぎに，データ伝送用のLEDの例を示します．図4.24(a)はフォト・カプラの1次側の駆動例です．とくに高速用でないフォト・カプラの場

図4.23　ダイナミック駆動時のブランキング期間　　**図4.24　データ伝送用LEDの駆動**

(a) フォト・カプラ1次側の駆動

(b) 周波数変調送信用回路

合，内蔵されているのは近赤外線LEDで，順方向電圧は1.5～2Vです．

同図(b)に示すのはディジタル信号により周波数変調（F1）を施して近赤外線を空間伝搬させるものです．この種のLEDはテレビのリモコンや煙報知機の光源として量産されており，発光率も高く，50mAの電流を流すと10mW程度の光出力が得られます．しかし，立上り・立下りの速度は0.5～1μsと遅いので，変調周波数は300kHzどまりでしょう．なお，図4.24(b)の送信部に対して，pinフォト・ダイオードを用いたFM受信部を組み合わせると，1200bps程度の信号を10～20m空間伝送できます．

例4-10 液晶表示器の駆動

液晶表示器は低消費電力で見やすく，多くの場所で使われようとしています．しかし，交流で駆動する必要があり，応答が遅くてダイナミック駆動がむずかしいなど，特有の問題がいくつかあります．

ここでは，液晶表示器の基本的な駆動方法を示します．

液晶表示器*（LCD）は消費電力がきわめて小さく，低い電圧で作動し，明るい場所でも見やすいなど，多くの利点をもち，時計，電卓，ハンドヘルド型パーソナル・コンピュータなど，電池動作の機械にはなくてはならない存在です．

一方，液晶自身は一種の電解質であり，温度変化や紫外線照射に弱いことのほかに，直流の印加ができない（電気分解して気泡を生じる）ことや応答速度が遅いためダイナミック駆動がむずかしいなどの駆動回路上の問題点があります．

また，液晶の用途は低消費電力が要求される場合が多いため，駆動回路も消費電力を小さくするよう工夫します．なお，液晶自体は発光しませんので，暗い場所では照明が必要です．

▶液晶のスタティック駆動回路　つぎに，実際の液晶駆動回路を示します．

図4.25(a)に示すのは基本的な液晶駆動回路です．液晶に直流分が加わらないよう，デューティ比50％の方形波を用いて両端をプッシュプルで駆動します．この周波数は液晶表示器にもよりますが，20～60Hzのものが普通です．点灯時に液晶に印加される電圧は，電源

* ここでは電界で作動するものをいう．

図 4.25 液晶のスタティック駆動回路

(a) 基本的な駆動回路　(b) 7セグメント表示用駆動回路

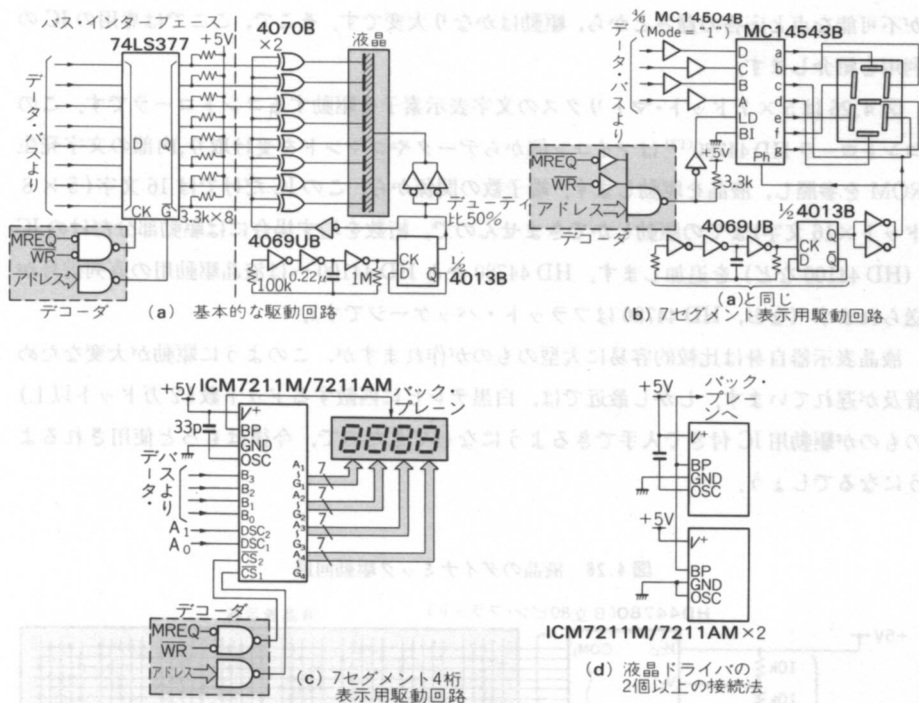
(c) 7セグメント4桁表示用駆動回路　(d) 液晶ドライバの2個以上の接続法

電圧と等しくなります($5\,V_{rms}$)．なお，TTL で液晶を駆動すると“H”レベルの不安定さのためうまくいきませんので念のため．

図 4.25(b)に示すのは7セグメントの数字表示用のものです．このICにはラッチ，デコーダ，Ex-OR ゲートが共存しているので，CPU のバスへの接続が容易です．ただしバスの“H”レベル電圧とホールド時間とに注意します．

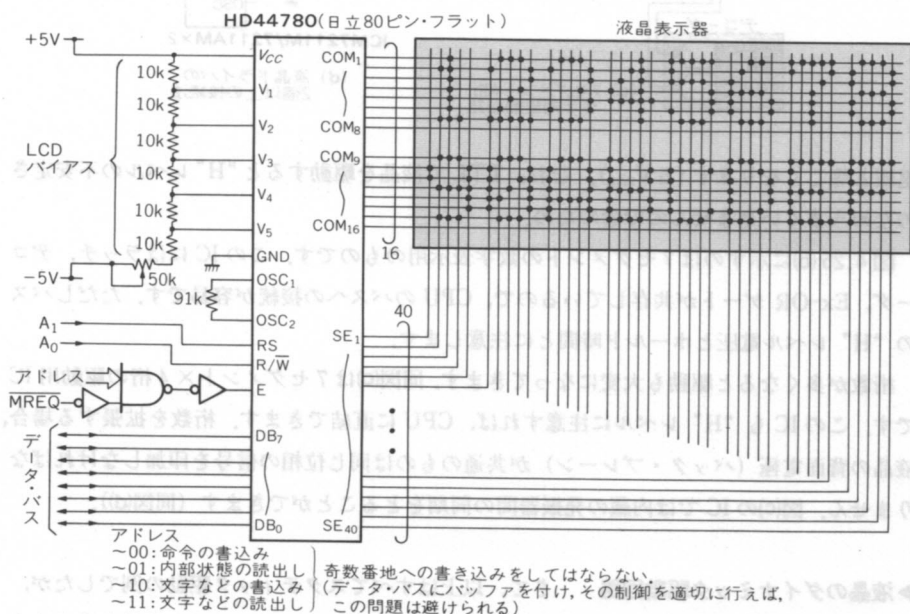
桁数が多くなると駆動も大変になってきます．同図(c)は7セグメント×4桁の駆動用ICです．このICも“H”レベルに注意すれば，CPU に直結できます．桁数を拡張する場合，液晶の背面電極（バック・プレーン）が共通のものは同じ位相の信号を印加しなければなりません．図(c)のICでは内蔵の発振器間の同期をとることができます（同図(d)）．

▶液晶のダイナミック駆動回路　さて，以上はすべてスタティック駆動の例でしたが，これ以上のセグメント数となると端子数だけでも大変なのでダイナミック駆動を考えます．

液晶のダイナミック駆動についてはいくつかの方式がありますが，前述のように直流印加が不可能な点と応答の遅さとから，駆動はかなり大変です．そこで，ここでは専用のICの利用を紹介します．

図4.26は5×7ドット・マトリクスの文字表示素子を駆動するコントローラです．このコントローラHD 44780[12]はマイコン側からデータやコマンドを受け取り，内部の文字発生ROMを参照し，液晶を駆動します．端子数の関係から，このICだけでは16文字(5×8ドット×16文字)までの駆動しかできませんので，桁数を増す場合には駆動部分だけのIC（HD 44100など）を追加します．HD 44780からHD 44100へは液晶駆動用の直列信号が送られます（なお，HD 44780はフラット・パッケージです）．

液晶表示器自身は比較的容易に大型のものが作れますが，このように駆動が大変なため普及が遅れています．しかし最近では，白黒テレビに匹敵するドット数(2万ドット以上)のものが駆動用IC付きで入手できるようになってきたので，今後はもっと使用されるようになるでしょう．

図4.26　液晶のダイナミック駆動回路

例 4-11　リレーの駆動

マイコンから高電圧・大電流の負荷をオン/オフしようとするとき，もっとも一般的なのはリレーを使うことです．ただし，リレーは誘導性負荷であり，駆動には注意が必要です．

一方，最近はラッチング型など各種のリレーが使われるようになってきました．これらの使い方についても触れることにします．

▶リレー駆動のための条件　出力インターフェースから見ると，リレーは誘導性負荷であって，ある程度電力も必要とします．そのため，つぎの項目を満足する必要があります．

(1) コイルに必要な電力を供給できること．

(2) 必要な速度で電流をオン/オフできること．

(3) 電流オフの際に生じるサージ電圧を吸収すること．

一方，出力としてのリレー接点には何らかの電気回路が接続されるわけで，そちら側からはつぎのような点が要求されます．

(4) 電力用としては，充分な接点容量（電圧，電流）をもつこと．

(5) 信号用としては安定な接点接触を保つこと（接触不良，チャタリング，接触電位などを含む）．

(6) 必要な速度での開閉が可能なこと．

普通，(4)，(5)，(6)から，まずリレーを選びます．

$$\frac{\text{リレーの接点で開閉できる電力}}{\text{コイルを駆動するのに必要な電力}\times\text{容積 (cm}^3\text{)}}$$

をリレー効率と呼びます(15)が，最近はきわめて高感度のリレーが現れ，リレー効率も1000以上におよんでいます．また，外形寸法も小型化され，125V 2Aの接点容量をもっていても16ピンDIPのICと同じくらいの平面形で間に合います．

一方，駆動回路を内蔵したリレーや，有極リレーにICを組み込んだ高速・高感度のリレーなども入手でき，リレーの選択も駆動回路の設計も，従来よりはるかに楽になってきました．

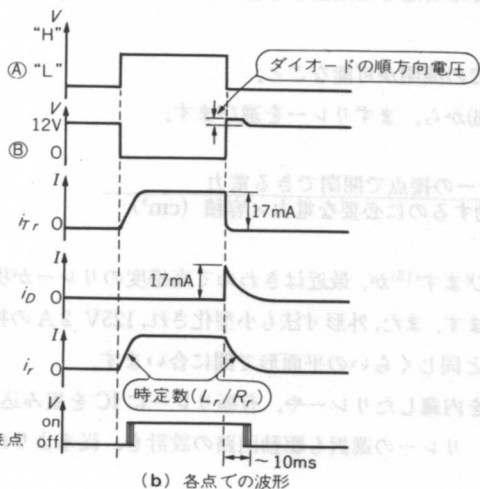
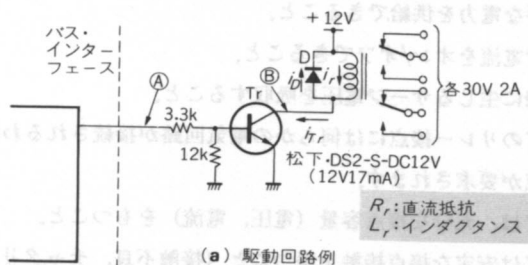
▶リレーの基本的駆動用回路　図4.27(a)に示すのは，30V 2Aの負荷を接続できるリレー駆動用インターフェースです．リレーは誘導性であり，そのままでは電流をオフした瞬間に高電圧を発生します．図中のダイオードDはそれを防ぐためのものです．同図(a)のTrをオフにすると，それまでリレーに流れていた電流は，そのままDに流れますから，ダイオードはそれに見合った順方向定格のものを使用します．

なお，同図(a)のようにたんに逆並列にダイオードを付加した場合，リレーをオフにしようとしても時定数L_r/R_rでしか電流が減衰しません．

これに対してオフ時間を速くするには，図4.28のような回路を使用します．同図(a)はダイオードに直列に抵抗R_1を挿入し，時定数を$L_r/(R_r+R_1)$としたもの，同図(b)は非線形素子(ここでは定電圧ダイオード)を挿入したものです．いずれもオフ時間を図4.27の数

図4.27　基本的なリレー駆動回路

(a) 駆動回路例

(b) 各点での波形

分の1にすることができます．

▶各種リレーの駆動回路　図4.29に示すのは微小信号負荷用としてリード・リレーを用いたものです．リード・リレーは微小信号に対しても安定ですが，接点の熱起電力が問題になるような箇所では発熱や接点間容量にも注意が必要です．また，リード・リレーの高速性を利用したい場合（リード・リレーは数100Hzでの開閉も可能）には駆動回路もそれなりに速度に注意します．

図4.30は有極（2巻線ラッチング）リレーの駆動回路です．以前は有極リレーは特殊な用途にしか用いられませんでしたが，最近では新たに高性能の小型有極リレーが開発されています．これらのリレーは高感度，高速で，常時電流を流しておく必要がないため低消費電力であり，自己保持作用があることから様々な用途が考えられます．

図4.30(a)は二つのコイル（巻線）を差動的に駆動する例ですが，*CR*を直列に挿入してあるので切換えの瞬間にはコンデンサの充電電流が短時間だけ流れます．定常状態では*R*

図4.28　オフ時間を速くする

(a)ダイオード，抵抗の利用　(b)非線形素子の利用

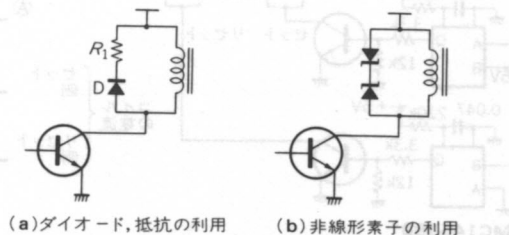



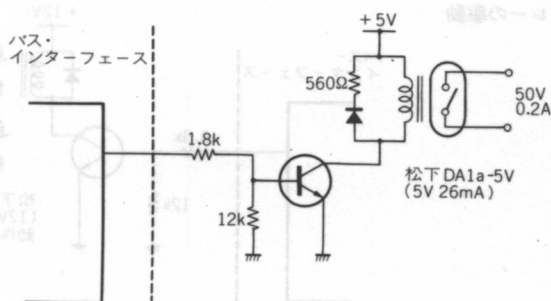
図4.29　リード・リレーの駆動

を通してわずかの電流が流れるだけです．この回路ではオン時に充分な電流・時間積が得られるように *CR* を選ぶことが大切です．一方，*C* が充電してしまわないと再切換えが不可能ですから，あまり早い周期でのオン/オフはできません．また同図(b)は，ワンショット回路を使ったもので，Ⓐ点のレベルにしたがってリレーがオン/オフします．いずれの場合も，ノイズでリレーがオン/オフしないように注意が必要です．

図 4.30　ラッチング型リレーの駆動

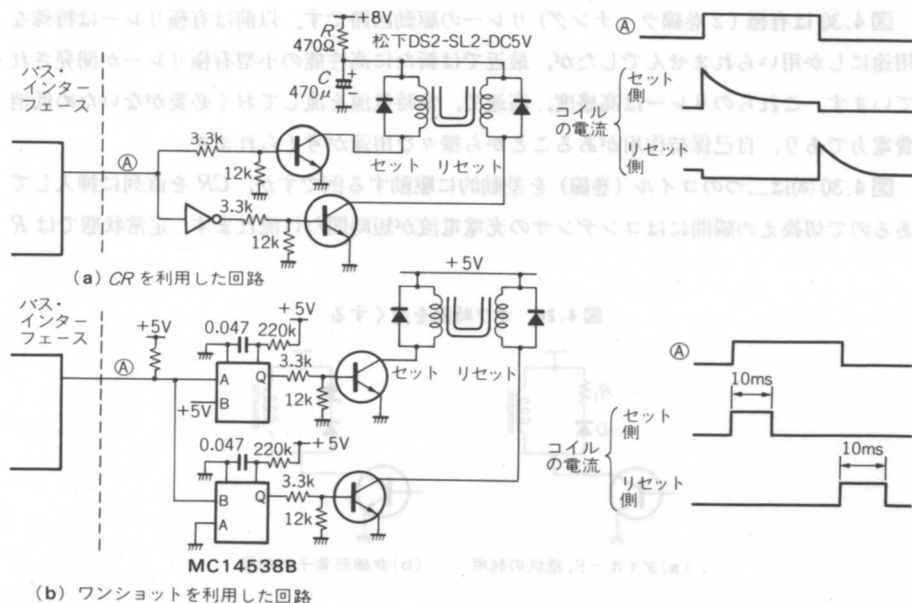



図 4.31　パワー・リレーの駆動

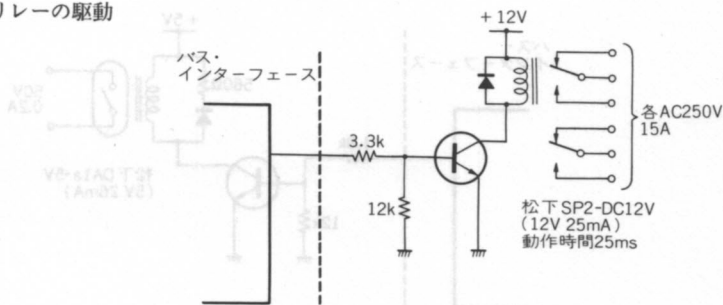
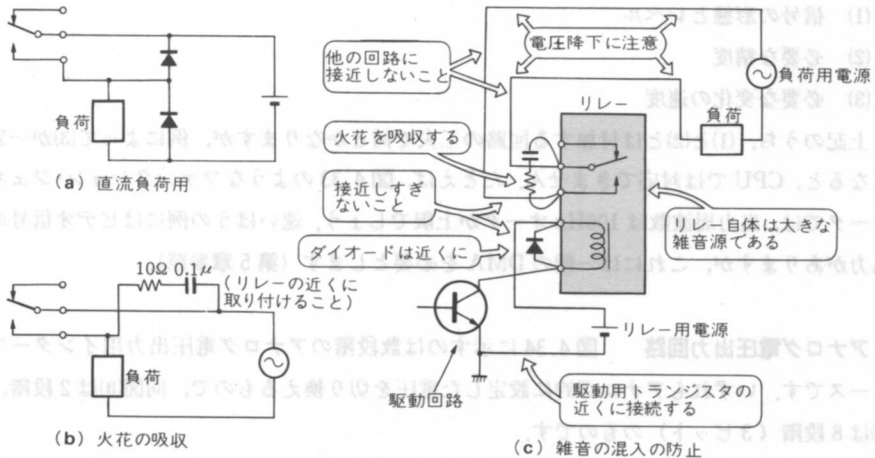



図 4.32 接点の保護と雑音の防止

▶パワー・リレーの駆動回路上の注意　図 4.31 に示すのは 250V 15A 級のパワー・リレーの例です．パワー・リレーになると駆動電流も多く必要なかわり，速度はあまり必要ではありません．ただし，大電流・高電圧の開閉による雑音などがマイコン側に影響を与えないように考慮しなければなりません．

リレーの負荷が直流の場合，図 4.32 (a)のようなダイオードにより，リレー接点を保護するとともに，火花による電磁波の発生を抑えます．同図(b)は交流でも直流でも効果があります．

リレー自体は技術の進歩により小型化され，パワー・リレーであっても基板に載るようになりましたが，逆にパワー関係の配線が他の回路に近接して新たな問題が生じています．図 4.32 (c)のような点を考慮して，インターフェースを介して雑音が混入するようなことがないよう注意すべきです．

例 4-12　アナログ信号の出力①――アナログ・スイッチによる方法

マイコンからアナログ信号を出力するには，一般には D-A 変換器を使用します．ただし，必要とする分解能が低くてよい場合，いろいろな手法が使えます．

ここでは，6 ビット以下の簡単なアナログ出力インターフェースの例をあげ，タイミングや絶縁についての考え方を説明します．

アナログ信号にも様々な種類がありますが，やはりつぎのような点に注目します．

(1)　信号の形態とレベル

(2)　必要な精度

(3)　必要な変化の速度

上記のうち，(1)と(2)とは付加する回路の工夫で何とかなりますが，例によって(3)が一定になると，CPUでは対応できません．たとえば，図4.33のようなファンクション・ジェネレータでは，出力周波数は100Hzオーダが上限でしょう．速いほうの例にはビデオ信号の出力がありますが，これには一種のDMAを必要とします（第5章参照）．

▶アナログ電圧出力回路　図4.34に示すのは数段階のアナログ電圧出力用インターフェースです．いずれもアナログ的に設定した電圧を切り換えるもので，同図(a)は2段階，(b)は8段階（3ビット）のものです．

図4.33　プログラマブルなファンクション・ジェネレータ

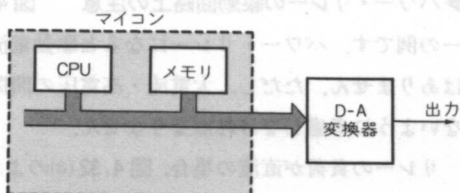
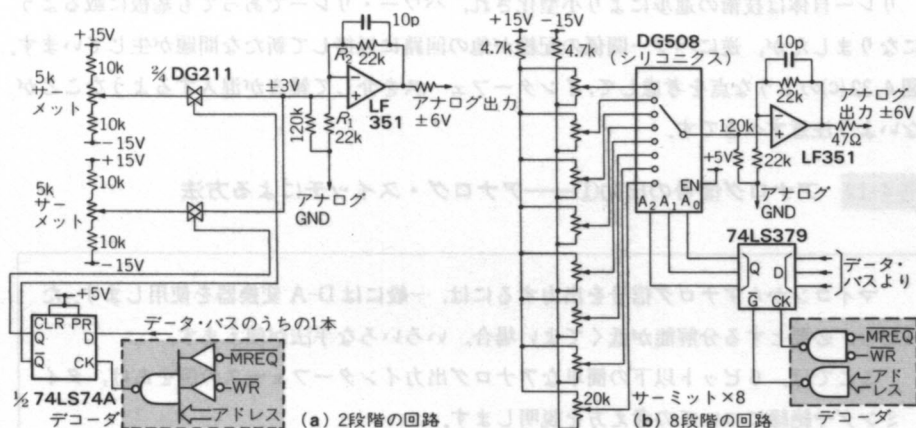



図4.34　数段階のアナログ出力用インターフェース

アナログ・スイッチは微小信号に対しても接触不良やチャタリングの心配がなく，速度も1μs前後と高速です．ただし，オン抵抗やオフ時の漏れ電流が存在します．同図(b)のDG 508[14]でも最大数100Ωのオン抵抗があるので，高インピーダンスで受けるようにします．

もう一つの問題は，切換え時のタイミングです．DG 508などのアナログ・マルチプレクサの多くはBBM（第3章コラムH参照）動作です．両方のスイッチがオフになっている期間は数100nsで，通常は問題ありませんが，とくに高速・低雑音が要求される場合は図4.35のように過渡状態にも注意が必要です．

一方，MBBのスイッチでは，スイッチ間のショートに注意しなければなりませんが，過渡状態では中間の電位が出力されます．

▶アナログ出力をマイコンから絶縁する回路　つぎに，アナログ信号をマイコン側と絶縁したい場合の例を図4.36に示します．

図4.36(a)はアナログ・スイッチのかわりにリレーを用いたもので，もっとも簡単です．ただし，リレーには接触不良，チャタリング，動作時間などの問題があります．図4.36(a)のDS型リレーでは，最小適用負荷が10μA 10mV（DC）となっています[15]．しかし，オン

図4.35　スイッチの切換えタイミング上の注意

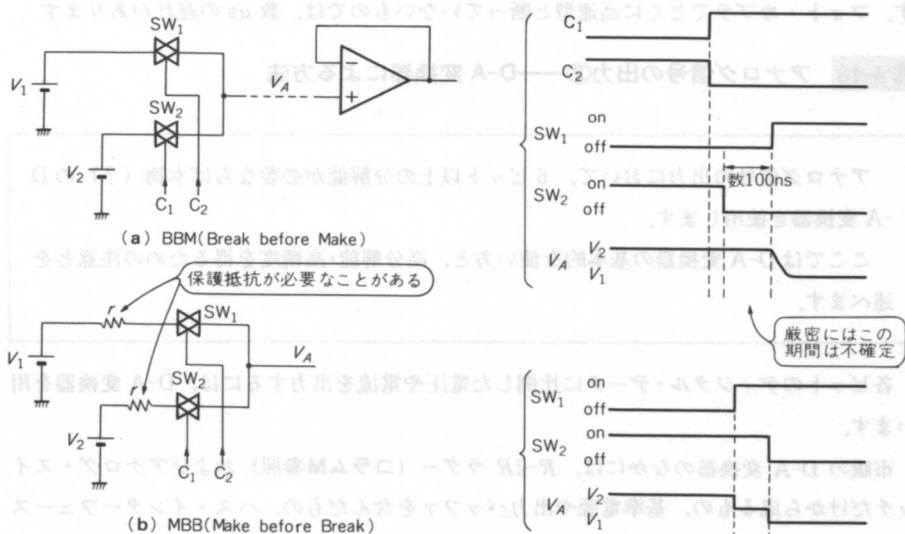
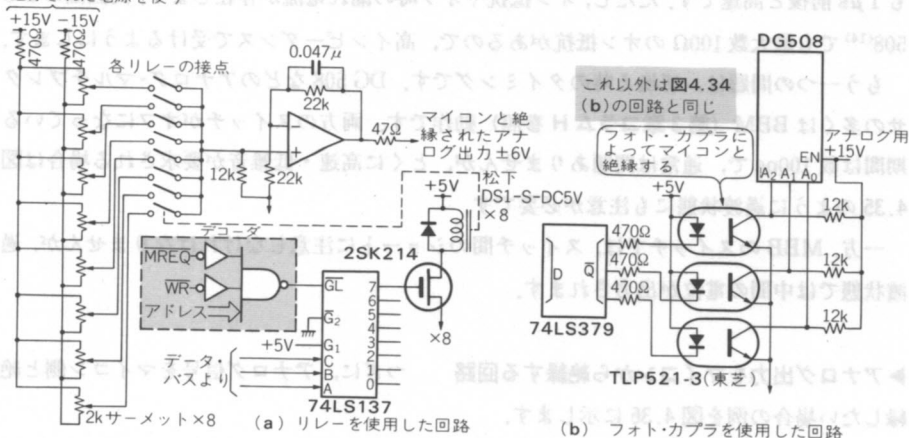



図 4.36　絶縁されたアナログ出力用インターフェース

（a）リレーを使用した回路　（b）フォト・カプラを使用した回路

抵抗はアナログ・スイッチと違って数10mΩですので，安定な開閉を望むには低インピーダンスで受けることです．この程度の小型リレーでは動作時間は数ms，リード・リレーでは1ms以下となります．チャタリングなどはこの時間内に収まると見て良いでしょう．

リレーでは動作時間が遅すぎる場合，同図(b)のようにフォト・カプラでの絶縁を考えます．フォト・カプラでとくに高速型と断っていないものでは，数μsの遅れがあります．

例 4-13　アナログ信号の出力②――D-A 変換器による方法

アナログ信号の出力において，6ビット以上の分解能が必要ならば本物（？）のD-A変換器を使用します．

ここではD-A変換器の基本的な使い方と，高分解能・高精度を得るための注意とを述べます．

各ビットのディジタル・データに比例した電圧や電流を出力するには，D-A 変換器を用います．

市販の D-A 変換器のなかには，*R*-2*R* ラダー（コラムM参照）およびアナログ・スイッチだけから成るもの，基準電源や出力バッファを含んだもの，バス・インターフェース回路を含んだものなど，多くの種類があります．

図 4.37　簡単な D-A 変換器

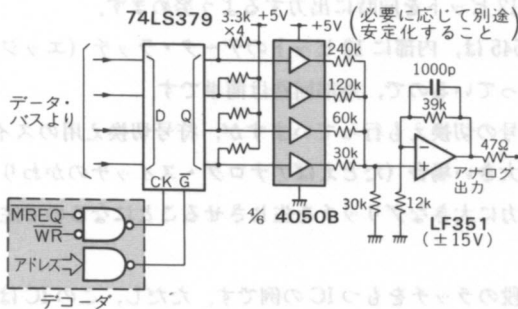
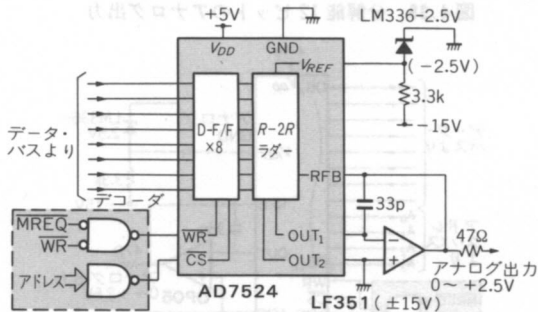



図 4.38　分解能 8 ビットのアナログ出力

▶ CMOS を利用した A-D 変換回路　　図 4.37 に示すのは非常に簡単な例です．CMOS のディジタル IC は出力が電源電圧いっぱいに振れるので，このような使用が可能です．ただし，出力インピーダンス（5 V 時に数 100Ω）の制限から，6 ビットが限界でしょう．

▶ D-A 変換器を使用した回路　　これ以上のビット数では本物（？）の D-A 変換器を使います．図 4.38 は 8 ビットの例です．アナログ・デバイセス社の AD 7524[6] は，AD 7523 に入力ラッチを設けたものと考えればよく，マイコン・バスへの接続が容易です．ただし，電源電圧が 5 V のときにしか TTL レベルとはならない点も 7523 のままなので注意が必要です．

ビット数がデータ・バスの幅（たとえば 8 ビット）を越える場合，データを同時に印加できるよう注意が必要です（例 4-4 を参照）．

図4.39に示すのは12ビットの例で，データ・バスを2回使うか，あるいはアドレスを使用するなどして，12ビットを同時に出力するよう努めます．

図4.39(a)のAD7545は，内部に12ビットのデータ・ラッチ（エッジ・トリガのDフリップフロップ）をもっているので，外部回路は簡単です．

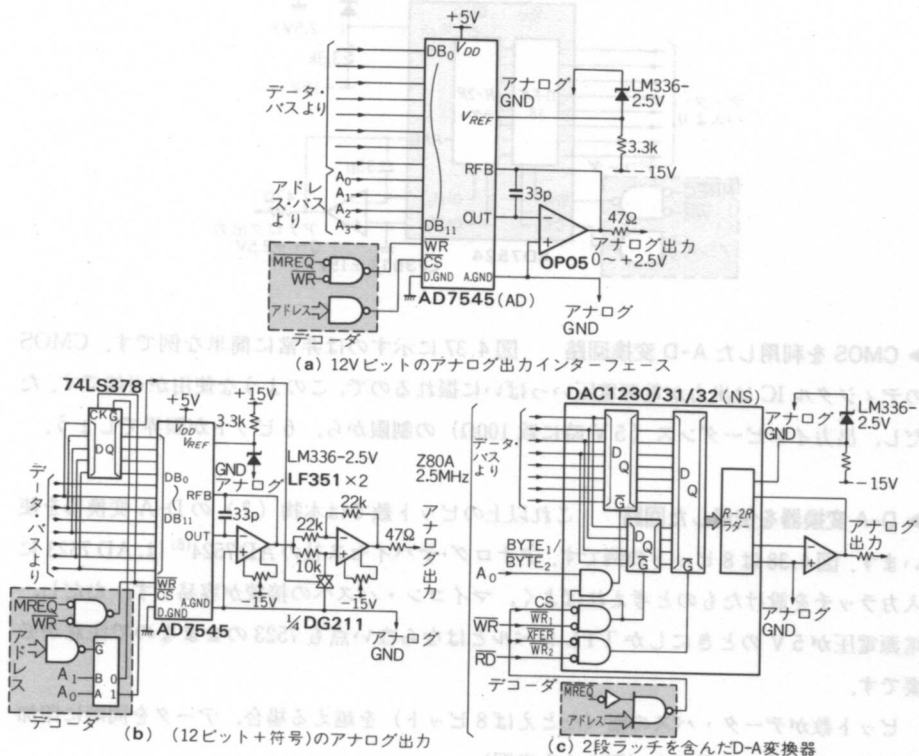
同図(b)は同時に符号の切換えも行っていますが，符号切換え用のスイッチとD-A変換器との速度の違いが大きい場合（たとえばアナログ・スイッチのかわりにリレーを使用する場合），アナログ出力に大きなグリッチを生じさせることになるので注意しなければなりません．

同図(c)は内部に2段のラッチをもつICの例です．ただし，このICはデータ・ホールド時間が90ns必要で，タイミング計算には手が抜けません．

図4.39　分解能12ビットのアナログ出力

(a) 12Vビットのアナログ出力インターフェース

(b)（12ビット＋符号)のアナログ出力

(c) 2段ラッチを含んだD-A変換器

▶必要な精度を得るための注意点　さて，技術の進歩により最近では16ビットのD-A変換器でも容易に入手できるようになってきました．しかし，D-A変換器が16ビットであっても，アナログ出力としてそれに相当する精度（広義の）が得られるわけではありません．そこで，実際によく使われる12ビット程度のD-A変換器で，必要な精度が得るための注意点をあげておきます．

⑴　**基準電圧源の温度特性**　D-A変換器によって基準電圧源を内蔵したものと外付けのものとがありますが，いずれにしても温度特性には充分な注意が必要です（例3-14参照）．

⑵　**アナログ・グラウンドを分離する**　ディジタル・グラウンドとアナログ・グラウン

図4.40　アナログ出力用インターフェースの構成例

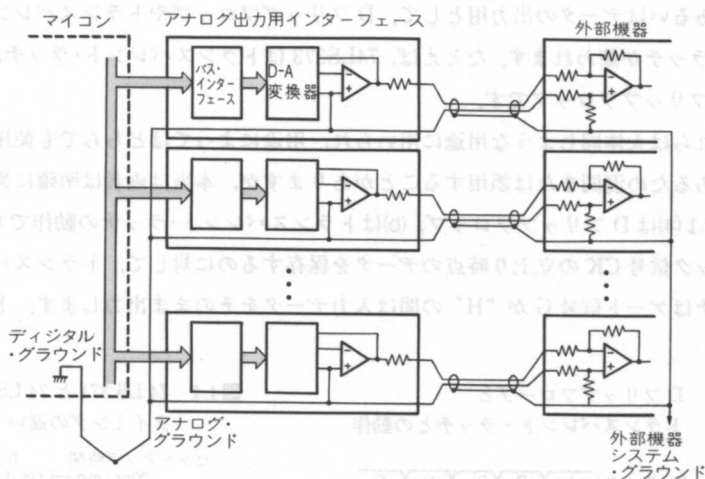
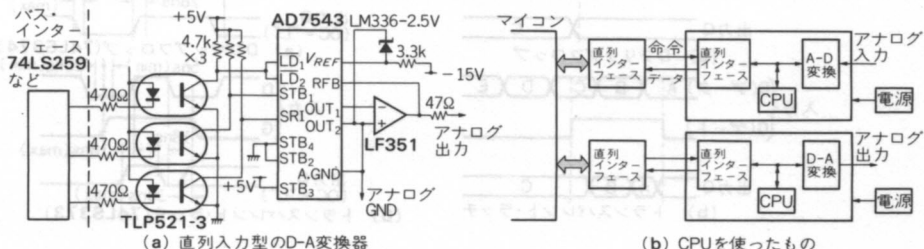



図4.41　直列転送を用いたアナログ出力用インターフェース

（a）直列入力型のD-A変換器　（b）CPUを使ったもの

ドを分離することはもちろん必要です．ただし，それだけでなく，各アナログ出力信号は外部機器側で差動で受けてもらうのが望ましいといえます（図4.40）.

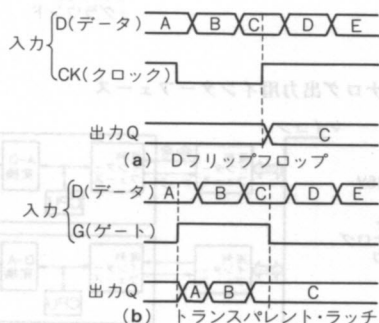
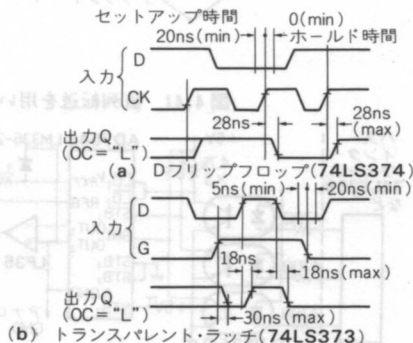
とくに，1台のマイコンにA-DやD-Aが何系統もぶら下がると，グラウンド間の電位差があちこちで生じます．これが許容できない場合は思いきって絶縁を考えます．その場合，バス全部をフォト・カプラで切ると本数が多いので，速度が遅くてよければ直列転送を考えます．図4.41(a)はデータを直列に入力するD-A変換器の例です．RS-232Cなどで結んだ図4.41(b)の構成も，マイコンが安くなった現在，決して実現困難ではありません．

コラムI　Dフリップフロップとトランスパレント・ラッチ

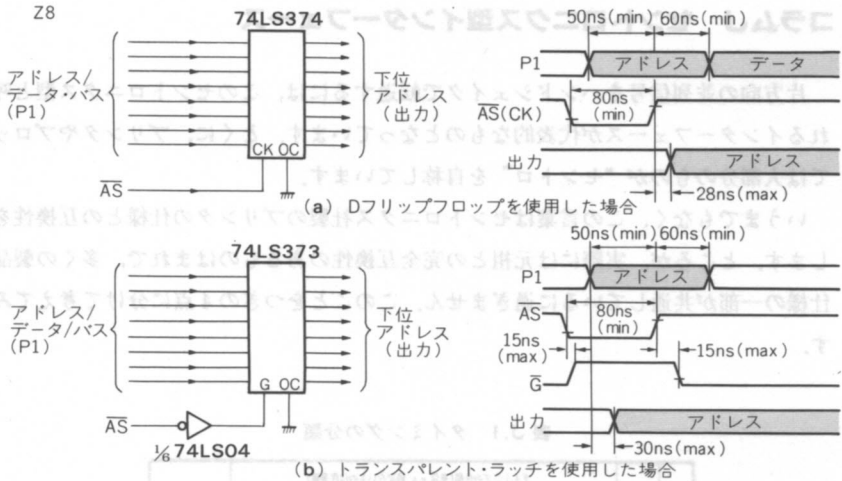
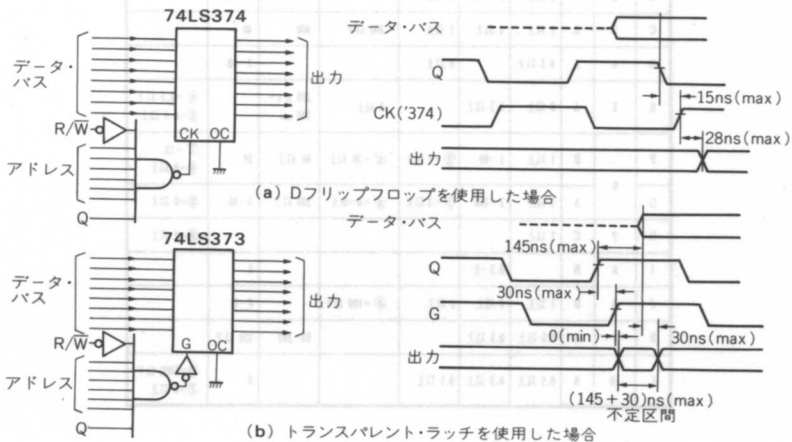
ストローブ信号によるデータのラッチ（固定）や，時分割バスからのアドレスの分離，あるいはデータの出力用として，Dフリップフロップやトランスパレント（透過型）・ラッチが使われます．たとえば，74LS373はトランスパレント・ラッチ，74LS374はDフリップフロップです．

これらは大体同じような用途に用いられ，用途によってはどちらでも使用可能な場合もあるため混同または誤用することがありますが，本当は両者は明確に異なります．

図I.1(a)はDフリップフロップ，(b)はトランスパレント・ラッチの動作です．前者がクロック信号CKの立上り時点のデータを保存するのに対して，トランスパレント・ラッチはゲート信号Gが“H”の間は入力データをそのまま出力します．トランスパ

図I.1　Dフリップフロップとトランスパレント・ラッチとの動作

図I.2　74 LS 374と74 LS 373とのタイミングの違い

レント(素通し)の名はここからついています．また図I.2に示すように，必要とするセットアップ時間やホールド時間が異なります．

つぎに，具体的な使用法を示します．図I.3は時分割バスからアドレスを分離するアドレス・ラッチ回路です．ここには，Dフリップフロップでもトランスパレント・ラ

図I.3　時分割バスからのアドレス分離回路

ッチでも使用可能ですが，後者のほうがアドレスが早く出力されます．

図I.4に示すのは6809の出力インターフェースです．第3章で示したように6809はE（データ・ストローブ）の立上り時点ではデータが確定していません．したがって，ここにトランスパレント・ラッチを使うと，データの書換え時にデータ不確定の期間を生じることがあります．

コラムJ　セントロニクス型インターフェース

片方向の並列信号をハンドシェイクで転送するには，このセントロニクス型と呼ばれるインターフェースが代表的なものとなっています．とくに，プリンタやプロッタでは大部分のものが“セントロ”を自称しています．

いうまでもなく，この言葉はセントロニクス社製のプリンタの仕様との互換性を示します．ところが，実際には元祖との完全互換性のあるものはまれで，多くの製品は仕様の一部が共通しているに過ぎません．このことをつぎの4点に分けて考えてみます．

表J.1　タイミングの分類

製品	メーカ	タイミングの仕様(単位 μs 特記ないものは代表値)							備考
		分類	①	②	③	④	⑤	⑥	
A	1	A	0.5 以上	0.5~200	0.5 以上		250 以上	12 以下	⑧=17 以上
B	2	A	1 以上	1 以上	1 以上			4 以上	
C	3	A	1 以上	2 以上	1 以上	160 以下	800	40	
D	4	A	0.5 以上		0 以上			2~10	
E	5	A	0 以上	0.5 以上		0 以上	300 μs~ 200 ms		④′=0.5 以下 ⑧=6.5 以上
F	6	B	1 以上	1~90	③′=1 以上	④′=30 以上	60 以上	10	⑦′=10, ⑧=0 以上
G		A	1 以上	1~100	③′=1 以上	④′=0~0.5	100 以上	5~10	⑧=0 以上
H	7	C	1 以上						⑨=2 以上
I	8	B		0.5~2				2	
J	9	B	1 以上	1 以上	1 以上	④′=100 以下		6~8	
K	1	A	0.5 以上	0.5 以上			64~183	154 以下	
L	10	A	0.5 以上	0.5 以上	0.5 以上			6	⑩=1000 以下 ⑦=0 以上

(1) コネクタやケーブルなどの機械的仕様

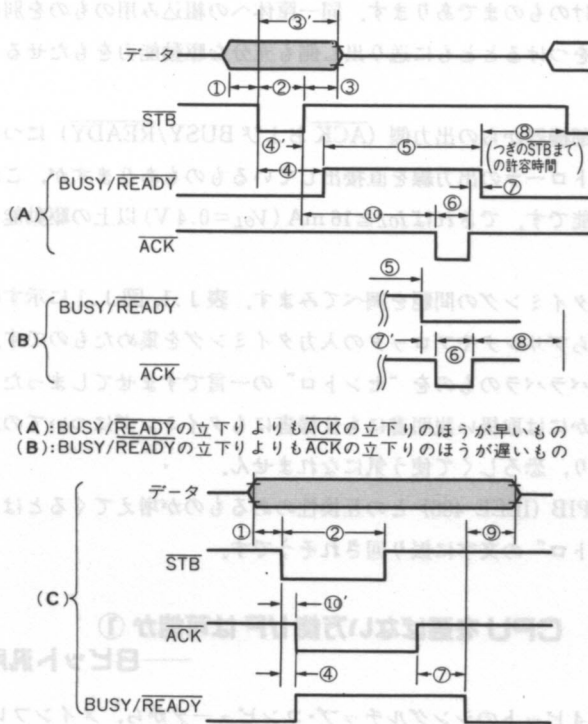
過半数の機器はセントロニクス社と同じDDK36Pのコネクタを採用しています．しかし，勝手なコネクタを付けても“セントロ”を自称しているものがあります．

ケーブルはすべてツイスト線を使うことになっていますが，実際には平行フラット・ケーブルが使われることが多いようです．短距離ならばこれでもかまいませんが，望ましくはありません．

しかし，コネクタやケーブルについては，違いが目に見えるものだけに，付属のものを使う覚悟さえあれば，問題となることは少ないといえます．

(2) 信号線の種類

データ（8ビット）とハンドシェイク用の信号（3本）とだけではなく，セントロニクス社のものにはリセット，紙送り，紙切れなどを示す信号が何本か付属していま

図J.1　プリンタなどのタイミング例

(A)：BUSY/READYの立下りよりもACKの立下りのほうが早いもの
(B)：BUSY/READYの立下りよりもACKの立下りのほうが遅いもの

す．これらのうち付属の信号については完全に同じであるものが少数派です．しかし，これもとくに問題とはなっていないようです．

(3) 制御用の符号について

印字，改行，改ページ，倍幅文字などの指令には，ASCIIのうち制御コード（00H～1FH)を使用しますが，これもまた，機器により違い，これについてはぜひともISOやCCITTの取り決めにしたがった適切な使い方を定めてほしいと思います．ただし，これも主としてソフトウェア上の問題ですから，本書では深入りしないことにします．

(4) 信号線の電気的仕様およびタイミング

本コラムで詳しく取り扱いたいのはこの点です．とくに重要なデータ線および3本のハンドシェイク線の直流特性とタイミングとに限って調べてみます．

まず，外部機器の入力側（データおよび$\overline{\mathrm{STB}}$)の直流特性では，シュミット入力のバッファにGPIB並の終端を施してあるものから，コントローラ(8041や8048系，あるいは3870などのF8系が多い）のMOS入力に10kΩくらいの軽いプルアップ抵抗をつけただけのものまであります．同一筐体への組込み用のものを別にして，ちゃんとした終端をつけるとともに送り出し側も充分な駆動能力をもたせることをおすすめします．

つぎに外部機器からの出力側（$\overline{\mathrm{ACK}}$およびBUSY/$\overline{\mathrm{READY}}$）については，ひどいものはコントローラの出力線を直接出しているものもありますが，これではマトモな終端は不可能です．できれば$I_{OL}\geqq 16$ mA ($V_{OL}=0.4$ V)以上の駆動能力がほしいところです．

最後に，タイミングの問題を調べてみます．**表J.1，図J.1**に示すのは手持ちの資料のなかからプリンタやプロッタの入力タイミングを集めたものです．

これだけバラバラのものを“セントロ”の一言ですませてしまったことにも感心しますが，なかには取扱い説明書にも仕様書にもタイミングについての記載が存在しない製品もあり，恐ろしくて使う気になれません．

今後はGPIB (IEEE-488) との互換性のあるものが増えてくるとは思いますが，当分は“セントロ”の文字に振り回されそうです．

コラムK CPUを選ばない万能I/Fは可能か ①
――8ビット汎用CPU

家電用の4ビットのシングルチップ・コンピュータから，メインフレーム用の32ビ

ットCPUまで，すべてに接続できるものを作るのは現実的ではありません．しかし，現在，制御用やパーソナル・コンピュータ用として使われている何種類かの代表的CPUについては，どのCPUにでも接続できるようなインターフェースを考慮しておけば非常に便利です．まず手始めに，8ビットの汎用CPU (Z80，6809，8088など)について考えてみます．

●データとアドレスとは多重化されているか？

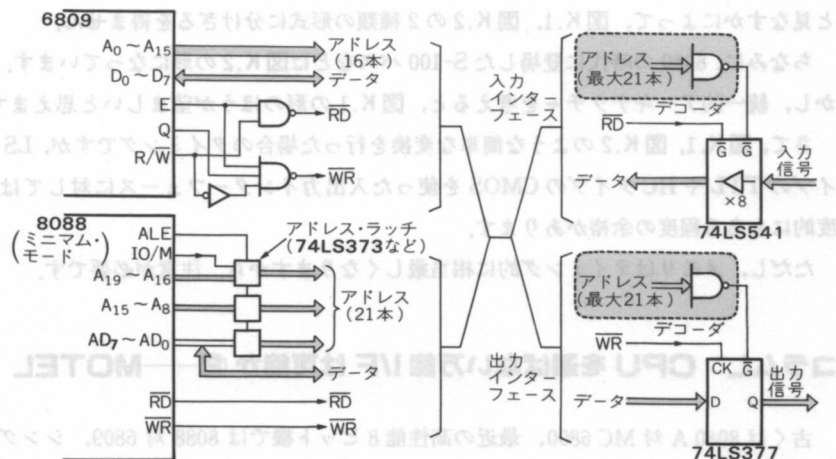
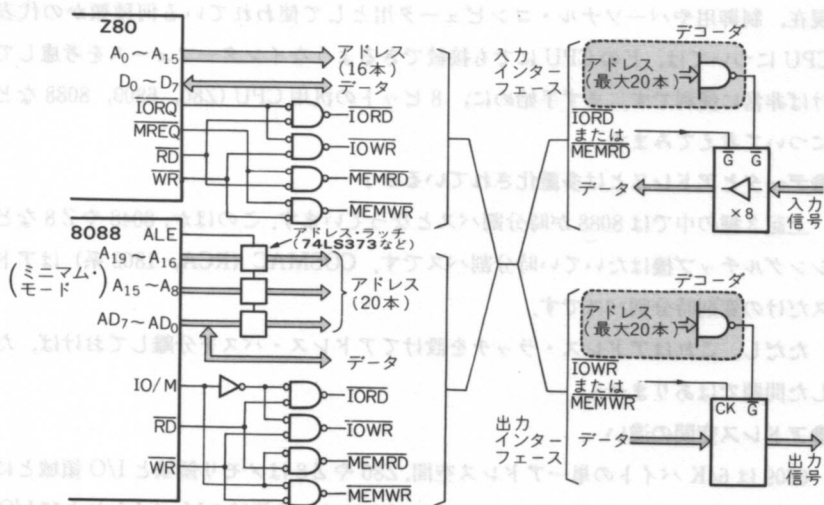
上記3種の中では8088が時分割バスとなっています．このほか，8048やZ8などのシングルチップ機はたいてい時分割バスです．COSMAC (RCA，1802系) はアドレスだけの変則時分割バスです．

ただし，これはアドレス・ラッチを設けてアドレス・バスを分離しておけば，たいした問題ではありません．

●アドレス空間の違い

6809は64Kバイトの単一アドレス空間，Z80やZ8はメモリ領域とI/O領域とに分かれた128Kバイト，8088 (ミニマム・モード) はメモリ領域1MバイトおよびI/O領域64Kバイトなど，CPUによってアドレス空間が異なります．

ここで非常に残念なのは，Z80のメモリ領域とI/O領域とを区別する信号が，他のアドレス信号と異なるタイミングで出力される点です (後述)．

図K.1　共通インターフェース──その1

図K.2　共通インターフェース――その2

●データ・ストローブ ($\overline{RD}$, $\overline{WR}$) の出力方の違い

6809はE (およびQ) およびR/$\overline{W}$, Z80や8088は$\overline{RD}$および$\overline{WR}$で，これは互いに変換可能です．他のCPUの多くも，このいずれかに分類できます．

ところが，先程の問題がかかわってきて話が面倒になっています．すなわち，メモリ領域とI/O領域を区別する信号を，アドレス線とみなすか，データ・ストローブ線と見なすかによって，図K.1，図K.2の2種類の形式に分けざるを得ません．

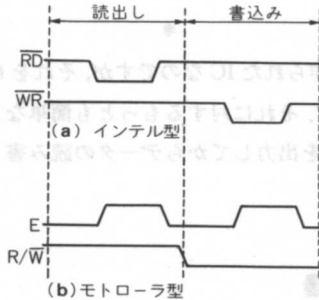
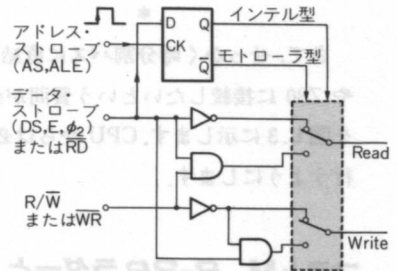
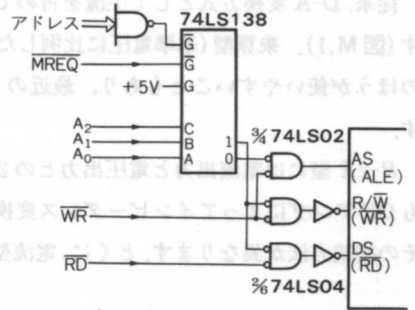
ちなみに，8080の時代に登場したS-100バスなどは図K.2の形になっています．しかし，統一的アーキテクチャを考えると，図K.1の形のほうが望ましいと思えます．

さて，図K.1，図K.2のような簡単な変換を行った場合のタイミングですが，LSタイプのTTLやHCタイプのCMOSを使った入出力インターフェースに対しては速度的にもある程度の余裕があります．

ただし，メモリはタイミング的に相当厳しくなりますから，注意が必要です．

コラムL　CPUを選ばない万能I/Fは可能か②――MOTEL

古くは8080 A対MC 6800，最近の高性能8ビット機では8088対6809，シングルチップ分野では8048対6805など，インテル社とモトローラ社はライバル関係を続け

図L.1　制御信号の違い

図L.2　MC 146818のMOTEL インターフェース部分

図L.3　時分割バス用のICを分離バスに接続する1例

ています．

ハードウェアの観点からは，インテル社が$\overline{RD}$と$\overline{WR}$との2本のデータ・ストローブ信号を使っているのに対して，モトローラ社はE(古くは ϕ_2)と R/$\overline{W}$とを使っています．これはシングルチップ機から32ビット機に至るまですべて同じです．

この両者を自動的に判別してしまおうというのが，“MOTEL”(Motorola-Intel)と呼ばれるインターフェースです．

図L.1に両者の信号を示しますが，バス・サイクルの開始点に注目すると，モトローラ型のE信号が必ず“L”であるのに対して，インテル型の$\overline{RD}$，$\overline{WR}$はともに必ず“H”であることがわかります．

図L.2に示すのはMC 146818のMOTELインターフェース部分です．このICはバス・サイクルの開始を判定するのにALE(アドレス・ストローブ)を使っているので，データとアドレスとが分離されている6809やZ 80には直結できません．しかし，6805

や8048などのほか，8085，8088，Z8などにも接続可能で，使用できる範囲はかなり広いといえます．

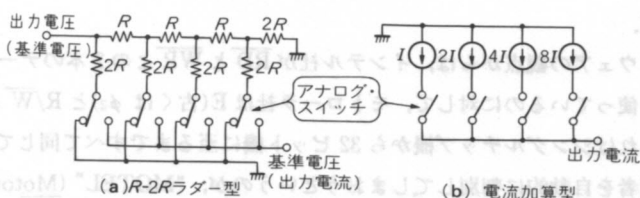
*　　*　　*

さて，せっかく時分割バスに直結できるように作られたICなのですが，それを6809やZ80に接続したいという質問が必ず出てきます．それに対するもっとも簡単な方法を図L.3に示します．CPUからは必要なアドレスを出力してからデータの読み書きを行うようにします．

コラムM　R-2Rラダーと D-A 変換器

従来，D-A変換方式として主流を占めていたのは，電流加算型とR-$2R$ ラダー型です（図M.1）．乗算型（基準電圧に比例した出力が出る）D-A変換器の場合，R-$2R$ 型のほうが使いやすいこともあり，最近のD-A変換器ではR-$2R$ 型を多く見かけます．

R-$2R$ 型には電流出力と電圧出力との2通りの使い方があります（図M.2）．いずれもOPアンプによってインピーダンス変換を行う必要がありますが，図に示すようにその接続方法が異なります．とくに，電流型では基準電圧とOPアンプ出力との極性が

図M.1　D-A変換方式

図M.2　R-$2R$ ラダーの2通りの使い方

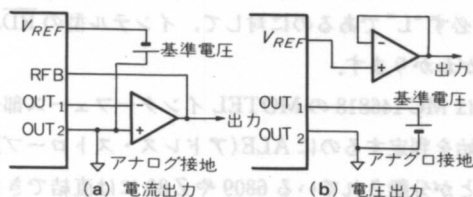
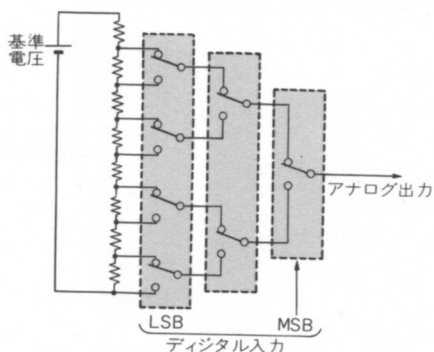



図 M.3　比例分割型 D-A 変換器の原理

逆になる点に注意して下さい．最近のCMOS構造のA-D変換器はどちらの使い方もできるものが多いのですが，内部のアナログ・スイッチの構造によっては，電流出力でしか使えないものもあります．

一方，最近のD-A変換器やA-D変換器(逐次比較型では内部にD-A変換器を持っている)のなかには，従来の R-$2R$ 型とは異なるものが現れました．図 M.3 に示したのはその例で，原理的には非常に簡単です．ただし，抵抗やアナログ・スイッチが多量に必要ですが，IC技術の進歩により，問題となってはいません．

これ以外にも，R-$2R$ 型と同じ構成で R-nR $(n \neq 2)$ 型としたものがあります．このような回路でD-A変換を行うと，ディジタル入力値とアナログ出力値とは比例しませんが，入力信号をROMに印加して値を変換して問題を解決しています．

第5章

高度な入出力インターフェースの設計

前章までは基本的な入力および出力インターフェースについて述べました．しかし，よく用いられるインターフェースのなかにも，第3章または第4章に書ききれないものが少なくありません．それらは，つぎのような理由によるものです．

(1) 映像信号のように，マイコンのソフトウェアでは対応しきれない高速のデータ転送を必要とするもの．通常，このような場合はDMA（直接メモリ・アクセス）を用いる．

(2) IEEE-488 (GPIB) のように，専用のインターフェース用LSIを用いることにより，容易に所定の仕様を実現できるもの．IEEE-488のほか，同期・非同期の直列通信回線（RS-232Cなど）もこれに含まれる．

以上のうち，映像信号は市販マイコンの標準的な出力として，またIEEE-488やRS-232Cは標準的な入出力として，ほとんどのものに装備されています．このほか，今後普及しそうな高速LAN（ローカル・エリア・ネットワーク；Ethernetはその代表的なもの）は，(1),(2)の両方に該当します．また，音声認識や音声合成もこの範ちゅうに入ります(図5.1)．

これらの処理のなかには以前はマイコンではまったく不可能なものもありましたが，マイコンのハード，ソフト両面の進歩により，あるいはアルゴリズム（手法）の進歩により，着実にマイコンで処理できる範囲が広がりつつあります．また，図5.1の範囲をさらに越える高次元の配列演算なども，机の上に乗るようなシステムで実現できるようになると思われます．

さて，余談はさておき，このような（ソフトウェアだけでは対抗できない）処理を行うには，つぎのような点を考慮する必要があります．

(1) 高速の処理にはDMA

汎用のCPU (Z80など) では，外部信号に対してもっとも速く応答できるのはDMAで

図 5.1　高度な入出力インターフェースの例

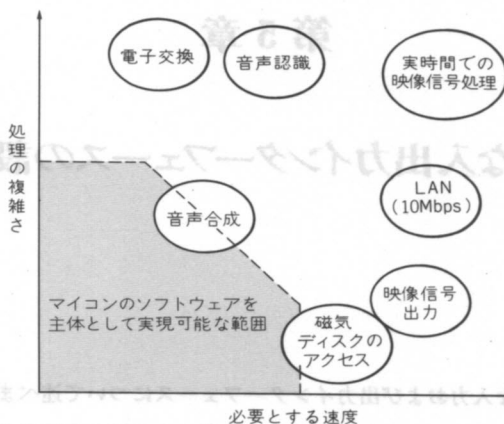
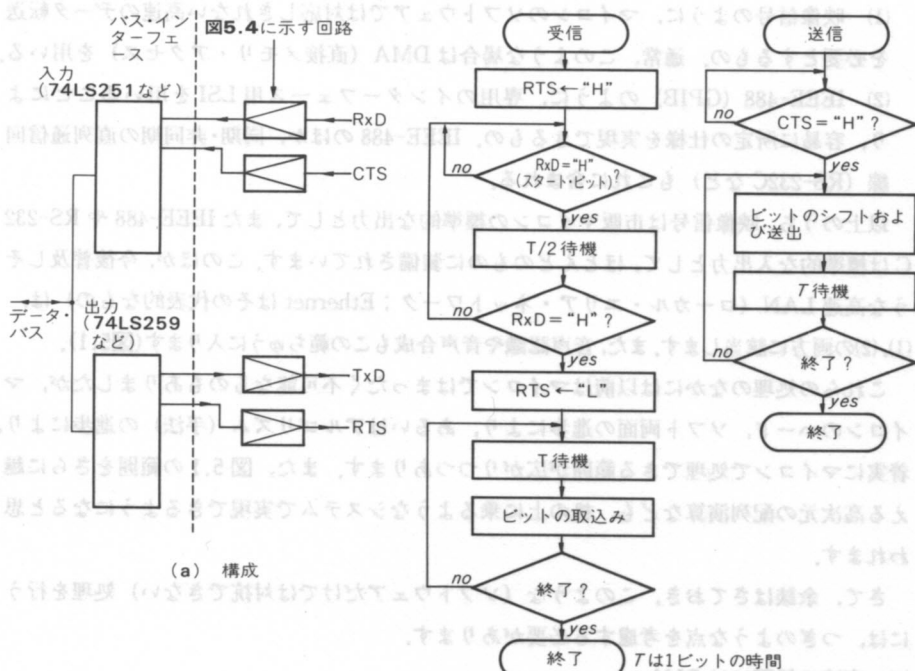
図 5.2　ソフトウェア制御による直列通信

バス・インターフェース
図5.4に示す回路
入力
(74LS251など)
RxD
CTS
データ・バス
出力
(74LS259など)
TxD
RTS

(a)　構成

受信
RTS←"H"
no
RxD="H"
(スタート・ビット)?
yes
T/2 待機
no
RxD="H"?
yes
RTS←"L"
T 待機
ビットの取込み
no
終了?
yes
終了

送信
no
CTS="H"?
yes
ビットのシフトおよび送出
T 待機
no
終了?
yes
終了

T は1ビットの時間

(RS-232Cのターミナル型の場合)

(b)　ソフトウェアの概要

す．ただし，サイクル・スチール*としない場合，CPU が行う通常の作業はストップします．

(2) 複雑な処理には専用 IC を使う

上記 DMA の制御のほか，各種の通信，映像信号の入出力，音声の認識や合成などの仕事には，専用の LSI が発表されています．可能でさえあれば，これらの IC を使うのが得策です．ただし，これらの IC を CPU に接続するには，両者の仕様を充分に検討する必要があります．

(3) 分散処理・並列処理を考える

どうしても処理が間に合わなければ，複数 CPU を使って分散処理または並列処理を行います（もっとも，前項の専用 IC の内部は，多くの場合シングルチップ・コンピュータとなっており，広い意味では分散処理といえる）．この場合，CPU 間のデータ転送方法が決め手となります．

本章では，このような入出力インターフェースからいくつか例をあげて説明します．

5.1 直列通信回線のインターフェース

例 5-1 直列通信回線

直列通信回線にも“たれ流し（ハンドシェイクを行わない）”式の低速のものから，10Mbps を越える LAN（ローカル・エリア・ネットワーク）までありますが，ここではマイコンに装備されることの多い非同期（または調歩同期）式のものを考えてみます．

図 5.2 に示すのは直列非同期通信をソフトウェアで行おうとするものです．図の例ではデータ（T_XD，R_XD）と CTS および RTS とだけを扱っていますが，これ以外の制御信号が必要ならば，それらもソフトウェアで処理します．同図(b)にソフトウェアの概要を示しますが，Z80A を 4 MHz で使用した場合，このような半二重的な使い方で 4800bps くらいまでは充分に実用になります．ただし，送受信を行っている間，CPU はそれに専念する必

* サイクル・スチールとは，CPU が行うバス・サイクルの間隔をぬって，外部機器から DMA を行うこと．

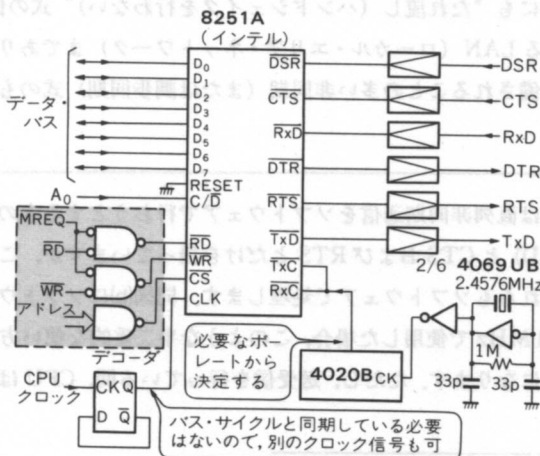
要があります．

図5.3に示すのは直列通信回線用のIC（インテル，8251A）[4]を使用したものです．8251Aは同期・非同期いずれのモードにも使用可能で，比較的広く使われています．なお，この手のICとしてはUART（ウェスタン・ディジタル社，1602タイプ）やMC6850/52（モトローラ社），Z80 SIOやDART（ザイログ社）などが使われていますが，LANが広まるにつれ，高機能のICが続々と登場しています．

なお，8251Aの$\overline{RTS}$，$\overline{DTR}$はたんなるビット単位の出力線，$\overline{DSR}$はビット単位の入力線ですから，名称のとおりに使用する必要はありません．

▶電流ループの例　つぎに，通信回線側とのインターフェースについて述べておきます．通信回線としては古くは電流ループが使われていました（図5.4(a)，(b)）．電流ループは古くさいように思われるかもしれませんが，低速（1200bps以下）での長距離通信には，雑音に強く費用が安いため，今でも使われています．同図(a)はよく使われる4線のものです．長距離を低費用で引っ張るのなら，同図(b)のように両端局に電源を設け，全部で2本の電線で引くのが望ましいといえます[1]．

▶ RS-232Cの例　図5.4(c)，(d)はRS-232Cの例です．(c)は専用のICを用いたもの，(d)

図5.3　専用ICを用いた直列通信回線

はOPアンプを用いたものです．RS-232Cは3kΩの受端インピーダンスに±3Vの電圧があればよく，短距離ならば(d)のOPアンプでも使用可能です．

さらに手を抜いたのが同図(e)で，これは相手側から送られてきた信号を整流して負側の電源としています．これは送り出し側の電圧が±9V以上で距離5m以内ならば使えないことはありませんが，本格的な使用には耐えませんので念のため．

RS-232Cの後継仕様としてRS-422/423が使われようとしています．これらについては

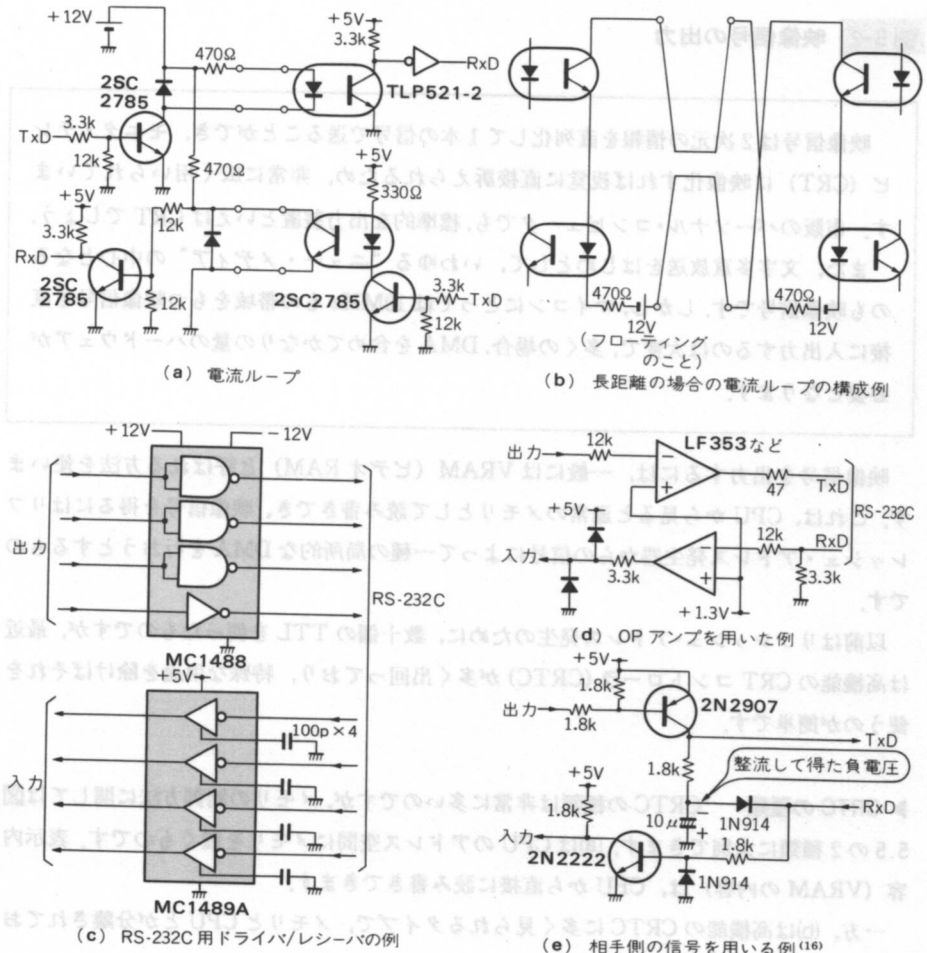
図5.4　通信回線用インターフェースの例

(a) 電流ループ

(b) 長距離の場合の電流ループの構成例

(c) RS-232C用ドライバ/レシーバの例

(d) OPアンプを用いた例

(e) 相手側の信号を用いる例(16)

専用のドライバやレシーバICを使うのが無難です(第3章,第4章を参照).

なお,RS-232Cなどの規格では電気的仕様のほか,コネクタの機械的仕様も細かく決まっています.使用される前に,参考文献などでよく調べること(たとえば『トランジスタ技術』,1983年12月号を参照)が必要です.

5.2 映像信号のインターフェース

例5-2 映像信号の出力

映像信号は2次元の情報を直列化して1本の信号で送ることができ,モニタ・テレビ(CRT)に映像化すれば視覚に直接訴えられるため,非常に広く用いられています.市販のパーソナル・コンピュータでも,標準的な出力装置といえばCRTでしょう.

また,文字多重放送をはじめとして,いわゆる"ニュー・メディア"の中心となるのも映像信号です.しかし,マイコンにとっては10MHzもの帯域をもつ映像信号を直接に入出力するのは大変で,多くの場合,DMAを含めてかなりの量のハードウェアが必要となります.

映像信号を出力するには,一般にはVRAM(ビデオRAM)と呼ばれる方法を使います.これは,CPUから見ると通常のメモリとして読み書きでき,映像信号を得るにはリフレッシュ・アドレス発生器からの信号によって一種の局所的なDMAを行おうとするものです.

以前はリフレッシュ・アドレス発生のために,数十個のTTLを使ったものですが,最近は高機能のCRTコントローラ(CRTC)が多く出回っており,特殊な用途を除けばそれを使うのが簡単です.

▶CRTCの種類 CRTCの種類は非常に多いのですが,メモリの制御方法に関しては図5.5の2種類に大別できます.(a)はCPUのアドレス空間にメモリを置くものです.表示内容(VRAMの内容)は,CPUから直接に読み書きできます.

一方,(b)は高機能のCRTCに多く見られるタイプで,メモリとCPUとが分離されてお

り，CPUからのメモリの読み書きはCRTCを介して間接的に行います．そのかわり，線分や円弧を引く仕事はCRTC側で面倒を見てくれます．

▶文字表示用とフル・グラフィック用　文字表示用でもフル・グラフィック用でも回路の構成は本質的には同じ（文字表示用では文字発生器（キャラクタ・ジェネレータ）が付く程度）ですが，メモリ量に差があります．

すなわち，文字表示用としては標準規格（後述）のCRTでは1画面あたり英数字2000文字程度が限界ですので，メモリも1画面あたり2KBあれば済みます．ところが，フル・グラフィック用では最低でも8～16KB，高度なものでは256KB～1MBのメモリを必要とします．普通の8～16ビットのCPUにとって，これだけのメモリをアドレス空間に置くのは負担が重いので，フル・グラフィック用としては図5.5(b)の構成が多く使われます．

図5.5 CRTCとメモリとの構成

(a) メモリをCPUのアドレス空間に置くもの

(b) メモリをCRTCが管理するもの

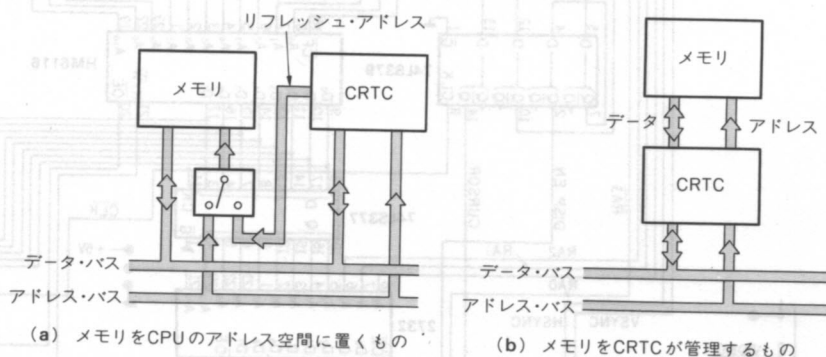



表5.1 映像信号のタイミング

水平周波数	15.75kHz
垂直周波数	60Hz
走査線数	525本*

＊飛越し走査を行った場合で，1フィールドあたり262.5本となる．飛越し走査をしない場合262本でもよい．

(a) NTSC規格

1行あたりの表示桁数	80	64	40
水平総文字数	130	109	65
ドット周波数	14.318MHz	12MHz	½×14.318MHz

1文字の構成…5×7ドット，字間2ドット
垂直周波数…60Hz
走査線数…262本（飛越し走査なし）

(b) 文字表示用のタイミング例

図 5.6　映像信号の出力

日立 HD46505S
DSPTMG
CUDSP
VSYNC
HSYNC
74LS158×3
74LS379
HM6116
74LS377
2732
74LS166
2/3 74LS10
74LS386
3/6 74LS04
1/2 74LS74A
DISP EN
CURSOR
RA3
RA2
RA1
RA0
CLK
出力
VIDEO
VD
HD
+5V
10k
D1
3.3k
2.2μ
16V
47Ω
14MHz
HC18U
33p
(セ：NPO)
1.2K
220p
(ス)

1/4 74LS00
100p
(ス)
120Ω
1/6 74LS367
(G1① pinはGND)
2/6 74LS367
MREQ·WR
MREQ·RD
2/6 74LS367
74LS139
+5V
1.2k
RM2
+5V
74LS623
×8
74LS266
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
アドレス・バス
1/6 74LS367
74LS623
D7
D6
D5
D4
D3
D2
D1
D0
データ・バス
CLK
2/4 74LS00
+5V
1/6 74LS04
74LS161A
*印 10kΩ **印 74LS04 ×2
(ス) スチロール (セ) セラミック (マ) マイラ

▶映像信号の規格　市販のパーソナル・コンピュータでは，高解像度用と銘打って特殊な規格の専用CRTを使うものがありますが，一般にはRS170など，既存の規格に合ったCRTを使うほうが良いでしょう．

RS170のタイミングは白黒かカラーか（NTSC），などにより細かく定まっていますが，単にモニタ・テレビ（CRT）に映すためならば，**表5.1**(a)（155ページ参照）を目安とすれば良いでしょう．明るさの階調をつけないのならば飛越し走査はしないほうが見やすく，走査線数は262本とします．

とくに，文字表示用を考えるとこの規格と1行あたりの文字数とから各タイミングは一意的に決まってしまいます．その例を**表5.1**(b)に示すので参考として下さい．

なお，CRTは多少のタイミング誤差（数%）があっても平気ですが，VTR（ビデオ・テープレコーダ）などのビデオ機器ではもっと厳密に規格通りの信号でないと作動しないことがありますので念のため．

▶文字表示用の映像信号出力インターフェース　**図5.6**（156，157ページ参照）に示すのは産業用の文字表示用インターフェース基板の例で，日立のCRTコントローラHD46505 S[11]を用いています．このICはE入力の立上りに対して$R/\overline{W}$信号にセットアップ時間が必要で，Z80など68系以外のCPUに接続するにはE信号を少し遅らせる必要があります．

例5-3　映像信号の入力

映像信号は非常に多くの情報を含んでおり，図形などを取り込むには有用です．しかしサンプリング周波数が10MHz程度になりますから，かなりハード的な処理が必要です．

映像信号を出力する装置としては，テレビ・カメラのほかに，VTRやビデオ・ディスク，あるいは放送受信用のチューナなどがあります．これらのなかには外部から同期をかけることが可能なものと，そうでないものとがあります．外部同期をかける場合と，そうでない場合とでは，処理の方法が若干異なります．

図5.7に示すのは，外部同期をかけた工業用TVカメラの出力を用いて，画面に映った白い物体の面積を求める回路です．

このような回路では同期信号発生用のICを使うのが簡単です．しかし，使い方や入手

が容易なものは多くありません. 図5.7で使ったソニーのCX2930Aは, 14.31818MHz (3.579545MHz×4)の水晶発振子を接続すればNTSC(PAL, PALM, SECAMも可)規格の同期信号を出力します. 電源も +5V で使いやすいのですが, パッケージがミニフラット型で試作に不便なのが欠点です.

画面の1ドット(画素)の縦横比を1:1にするには, 走査線数262.5本に対してドット・クロック周波数を約6MHzとします. ここでは, 74LS132によるヒステリシス発振器

図5.7 面積測定回路

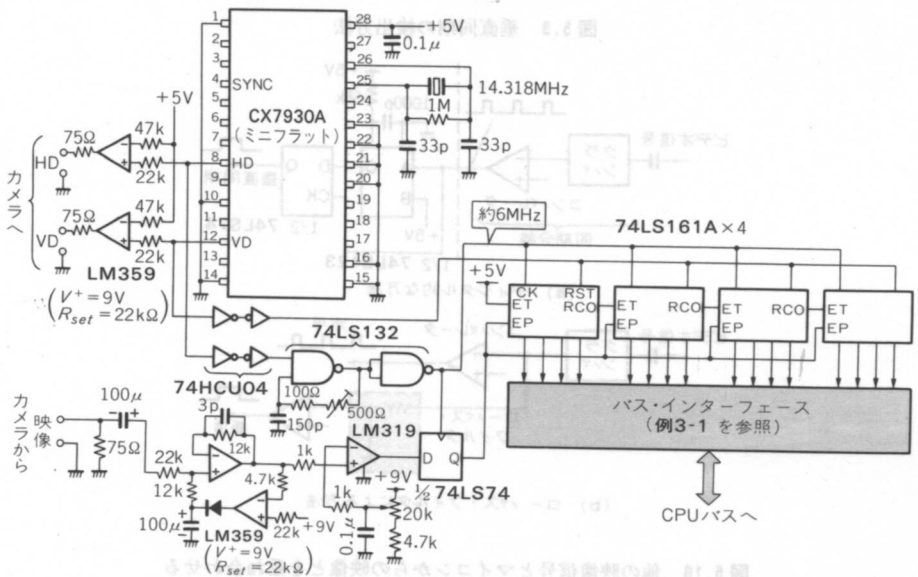



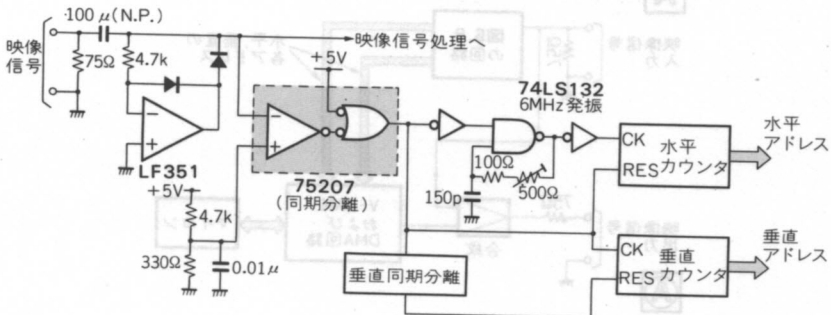
図5.8 映像信号から水平・垂直のアドレスを作る方法

を使っていますが，電源電圧や温度が変化すると，周波数が最大で数%変動します．もし高い精度を必要とするならば，PLL を使って 14.318MHz と同期させるのがよいでしょう．

図5.8に示すのは，外部同期をかけられない装置からの映像信号の入力方法です．この場合は，映像信号から同期分離を行い，そのなかから垂直同期信号を抽出し，場合によっては飛越し走査の各フィールドを区別しなければなりません．

図5.9(a)に示した垂直同期検出回路はディジタル的なもので，同図(b)のローパス・フィ

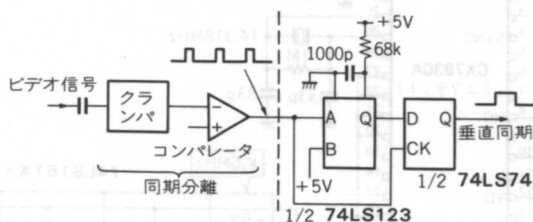
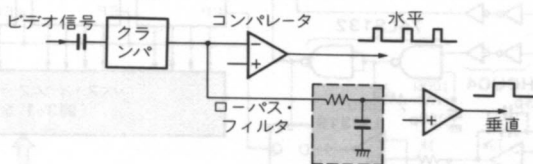



図5.9　垂直同期の検出方法

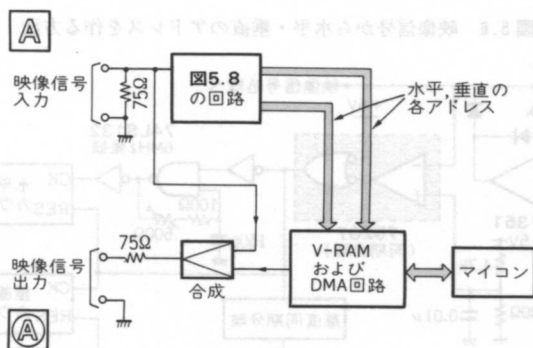
図5.10　他の映像信号とマイコンからの映像とを重ね合わせる

ルタ型のものに比べて簡単で確実です．水平のドット・クロックを与えられた映像信号に追従させるには，PLL が必要となります．

図 5.8 の回路を用いて図 5.10 のような構成とすれば，外部の映像信号とマイコンで作った画面とを重ねることができます．画面を重ねる機能を含んだ CRT コントローラ用 IC もありますが，図 5.10 の構成とすればドット数などが好きなように設計できます．

5.3 分散処理のインターフェース

例 5-4 分散処理（複数 CPU システム）

マイコンにとってもっとも苦手な高速かつ並列的な処理を行うには，複数 CPU システムを考慮します．CPU の価格が非常に安くなった今日，仕様（部品数など）が許すのなら，仕事を細かく分割したほうがシステム構成はむしろ単純です．

ここでは，簡単な例として共通バスをもたない複数 CPU システムの CPU 間のやりとりについて説明します．

たとえば，DMA を用いて映像信号を出力する場合，実効的な処理速度は半分程度に低下します．とくに，外部からの割込みなどに対する応答速度が悪化し，ただでさえ不足がちなマイコンの（見かけの）速度を一層低下させています．これを防ぐ手段はいくつかありますが*，ここでは複数 CPU システムを考えます．

図 5.11 に示すのは分散処理を大幅に取り入れたシステムの概念図です．これは複数 CPU システムとしては非常に簡単な構成で，とくにデータの流れが単純なため，ハードウェアの設計も容易です．

図 5.12 に示すのは実際の例で，入力処理部は外部クロックに応じ，20μs 以内に外部からのアナログ信号を取り込む必要があります．マイコン（ホスト）側はそのデータを処理するとともに，画面にデータを表示します．

低価格帯の市販パーソナル・コンピュータでは，DMA オーバヘッドが加わり，20μs の応答時間は実現できません．また，ハードウェアだけで構成するのも結構大変です．そこ

---

* 画面用のバッファ・メモリを設け，実質的な DMA オーバヘッドを下げることが多い．

でこの例では，スレーブCPUはデータの入力だけを受け持ちます．

2個のCPUのデータ転送は，同一のメモリに対して両方のCPUから同時に読み書きを行う可能性がなければ比較的簡単です．

図5.13(a)に示すのはもっとも簡単なもので，一方からは出力ポート，他方からは入力ポートと見なせます．双方向のやりとりが必要な場合，これを2組設けます．この回路では，

図5.11　分散処理化を図ったシステムの例(各CPUのバスは共通でないもの)

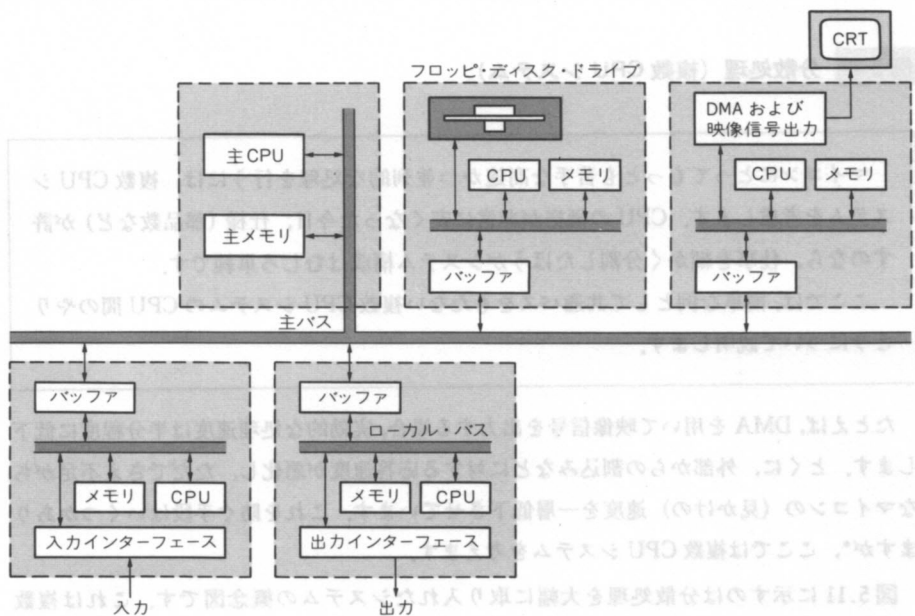



図5.12　複数のCPUを使った簡単なアナログ入力インターフェース

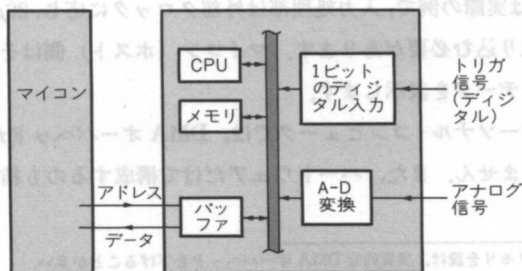
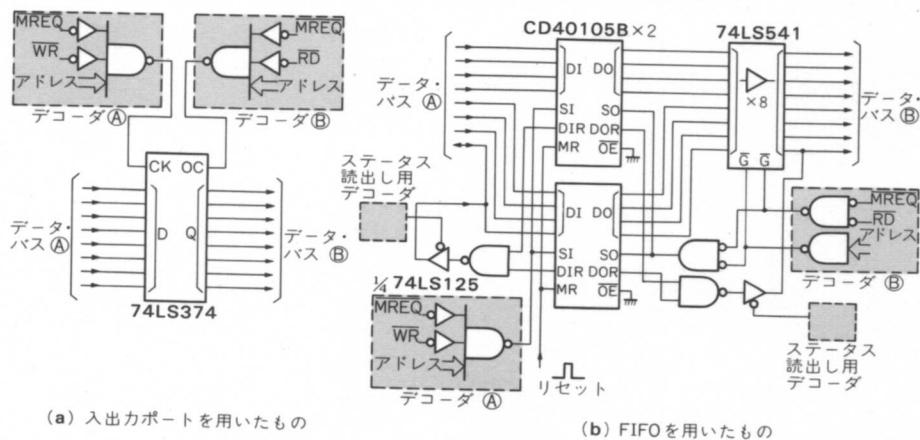



図 5.13　複数 CPU 間のデータのやりとり (一方向のもの)

(a) 入出力ポートを用いたもの

(b) FIFOを用いたもの

書き込み中に他方からデータを読み出した場合，データが保証されませんから注意が必要です.

同図(b)に示すのは FIFO[9] を使ったものです. この回路の利点は, 読出し側と書込み側とがまったく非同期的に作動してもデータが失われないことです. ただし, FIFO が満杯になればダメなので念のため.

より複雑な分散処理を行う場合, バスを共有することになります. この場合は各 CPU のバス使用タイミングを調停する手段が必要となります. 本書ではこれには触れませんが, バス調停用の IC も作られており, 今後はそのようなシステムも増えると思います.

第6章

入出力インターフェース・ユニットの実際

第3章～第5章では個々の入出力インターフェースの設計法を説明しました．実際のシステムでは，以上の方法をもとにして必要とされる入出力インターフェースを設計すれば良いのですが，似たようなインターフェース回路を毎回設計するのも大変です．そこでメーカの多くは各種の標準ユニットを用意しており，必要に応じてそのユニットを組み合わせ，システムが構成できるようにしています．

これらのユニットの中には，

(1) 特定のパーソナル・コンピュータの拡張スロットに差し込むためのもの．

(2) 業界標準規格に従っていて，他社のものとも任意の組合せが可能なもの．

(3) 同一シリーズの基板との組合せしか考慮していないもの．

などがあり，基板やコネクタの形状，信号線の電気的な規格などが，各々に定められています．

ここで紹介するのは上記(3)に含まれるものです．このシリーズの基板の特徴としては次のような項目があげられます．

(1) Z80 CPUを使い，実績のある（いわゆる“枯れた”）ハードウェア/ソフトウェアを利用できる．

(2) ソフトウェアの負担をなるべく少なくするよう考慮してある．

(3) 入出力の多くはメモリ・マップとなっており，割込みは通常は使わない．

これらの点から，特定のパーソナル・コンピュータ用の入出力モジュールなどとは設計方針が異なっていますが，現場での入出力インターフェースとして参考になる部分が多いでしょう．

表6.1 ユニット基板の1例

分類	内容
CPU基板	CPU,メモリ,デバッグに必要な入出力など
入出力基板	絶縁されたディジタル入出力(各16本)
	アナログ入力(8ビット×8チャネル)
	アナログ出力(12ビット×4チャネル)+ディジタル入出力
	スイッチ読込み用(16回路×16接点)
	遠隔操作用(スイッチ,LED,ポテンショメータなど)
	その他数種類(表示用など)
メモリ基板	28p スタティック・メモリ×8個 (ROM/RAM兼用,最大64Kバイト)
スレーブCPU基板	直流モータ制御用の入出力を持つスレーブZ80
その他	映像信号の取込み用,VRAMなど,数種類

●全体の構成

このシリーズに含まれる基板の1例を表6.1に示します.一般に,ロボット制御などのシーケンサ的な応用では,リレーやスイッチ類,あるいはオープン・コレクタの入出力が大半を占めます.それ以外では直流サーボ用のアナログ出力と,表示関係(LED,液晶,またはCRTモニタ)出力とがあれば,ほとんどの場合に間に合います.

一方,プロセス制御に使う場合はアナログ入力が重要となります.これは低速・高分解能(12~14ビット,毎秒10回程度)と,高速・低分解能(8~10ビット,毎秒10,000回程度)との2種類が必要です.

また,最近は工業用テレビ・カメラ(ITVカメラ)やイメージ・センサを使う機会も多く,そのためのインターフェースも考慮してあります.

このようにユニットを組み合わせて色々なシステムに応用しようとする場合,何があれば必要かつ十分であるか,良く検討することが重要です.

6.1 CPU基板

CPU基板(付図6.1に全体の回路図を示す)はすべての装置に少なくとも1枚は使用され,大半の用途では"心臓部"となります.したがって,CPU基板の設計は,このようなユニット基板のシリーズを構成する際,非常に重要となってきます.

●CPU基板の特徴

CPU基板の中心はいうまでもなくCPUです．このほか，一般にはクロック発振回路，リセット回路，バス・バッファなどが必要となります．

本来，このようなユニットのシリーズでは，メモリや入出力インターフェースを別々の基板とし，用途に応じてそれを組み合わせるような使われ方をします．したがって，CPU基板にはメモリや入出力インターフェースは載せなくても構いません．

しかし，どのような用途でも多少のメモリは必要（RAMなしのシステムはあり得るが，ROMなしは一般にはあり得ない）ですし，最低限の標準的な入出力インターフェースはデバッグや小規模の用途に便利です．このため，この基板にはメモリと若干の入出力インターフェースが設けられています．ただし，これらはデバッグや割込みなど特殊な用途に使われることを主眼としており，他のインターフェース基板とは設計が若干異なる点に注意して下さい．

●CPUの選定

このシリーズは，シーケンス・コントローラ的な用途，あるいはプロセス制御用などを主眼としています．これらの用途では処理速度はさほど重要ではありません．むしろ，簡単かつ確実なハード/ソフトで動作することが要求されます．

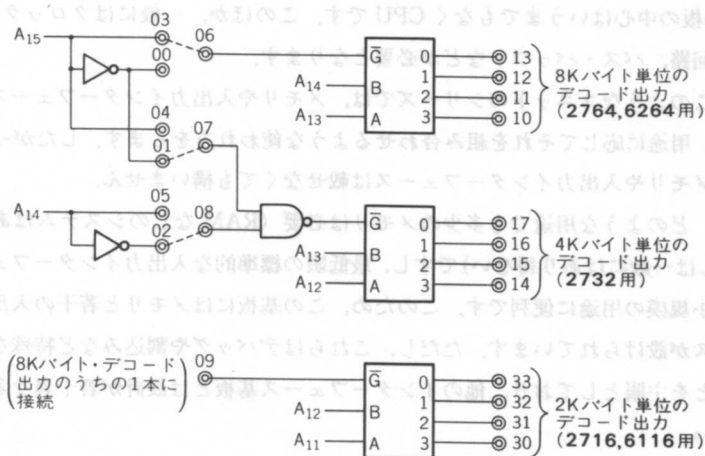
この基板の設計時点（1982年）ではハード/ソフトともに使用実績があり，かつ開発環境の整ったCPUとしてはZ80，8085，6802などの汎用8ビット機しかありませんでしたので，その中からZ80を使用しました．その後，Z80は（その高速版Z80A，Z80Bを含む）各社で量産され，安定かつ安価に供給されました．

しかし，これからCPU基板を設計するのなら，シングルチップ・コンピュータ（8048/51，Z8，6301など）を使って部品数を減らすか，より高機能のCPU（8088，6809など）を使って性能を稼ぐかを選ぶ必要があるでしょう．

●メモリ

CPU基板にメモリを載せるか否かは意見の分かれるところです．しかし，ROMやRAMが各々2Kバイトずつもあれば充分かつ簡単な用途は少なくありません（むしろ，大半の用途がその程度だといえる）．そのような場合にCPU基板単体でも用がすむよう，この基板には28ピンのスタティック・メモリ用のソケットが4個用意してあります．

設計時点ではROMは2716（2Kバイト）や2732（4Kバイト）が，RAMは6116（2

図6.1　アドレス・デコーダの構成

Kバイト）が全盛で，この基板に実装できるメモリも2716×2＋6116×1（ソケット1個は予備）の合計6Kバイトといったところでした．しかし，メモリの進歩は早く，1984年の段階ではROM，RAMともに8Kバイト型（2764，6264）が主流となり，CPU基板上に24Kバイト（1個は予備）が実装できるようになりました．これはかなりの量で，ほとんどの用途でメモリ基板は不用となっています．

なお，使用するメモリ（ROM 2716/2732/2764，RAM 6116/6264の各型）およびそのアドレスは，ジャンパ（ショート・ピン）で選択できるようになっています（図6.1）．

●直列通信回線（RS-232Cインターフェース）

この基板には8251Aを中心としたRS-232Cインターフェースが設けられています（図6.2）．実際の応用では直列通信回路を使わない場合もありますが，デバッグの際，このような標準的なインターフェースがあると標準のモニタ・プログラムが使用でき，非常に便利です（ハード/ソフトのデバッグの手順については巻末も参照して下さい）．

図からおわかりのように，この基板では8251AのクロックとCPUのそれとが共通になっています．これは，設計が古くクロックとして2.5MHz（ボーレートの関係から2.4576MHzを使うことが多いが）を主眼としていたためです．CPUにZ80Aを使い，クロックを4MHzにするには，2分周した2MHzのクロックを8251A（およびZ80A CTC）に与えるのが適当です（図6.3を参照）．

また，ボーレート用のクロックはZ80 CTCから，8251AのT_xC，R_xCに与えていますが，この信号はフリップフロップ（74LS74A）で2分周してあります．これは，Z80 CTCの出力が細いパルス状であってそのままでは8251Aのタイミング規格を満足しないためです．

図6.2 RS-232Cインターフェース部分

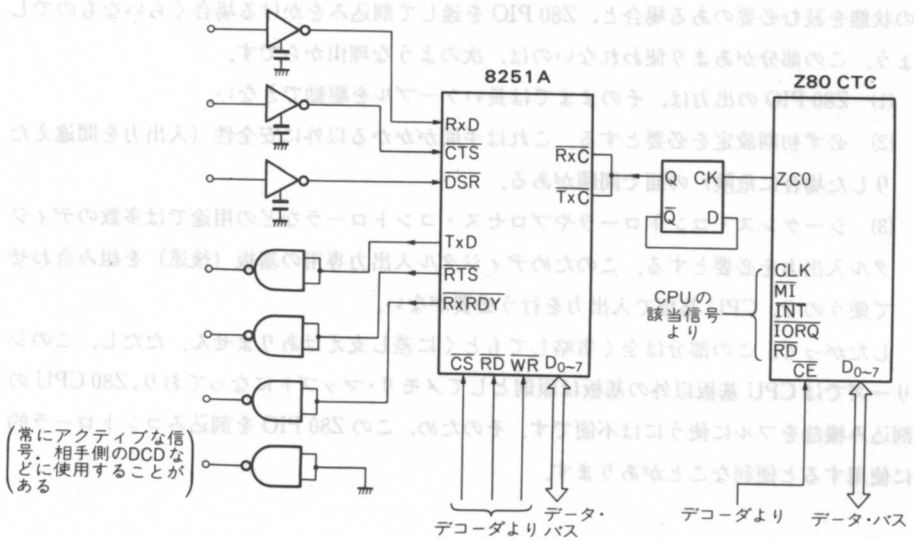
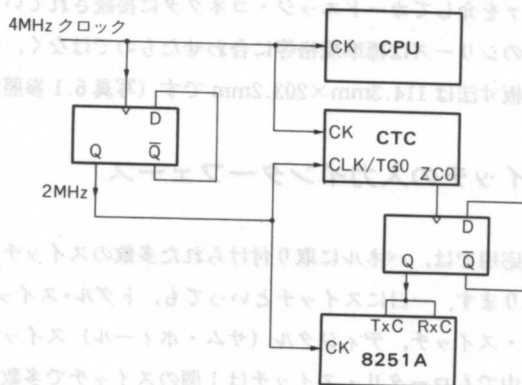



図6.3 クロック4MHzでの変更点

RS-232C 用のレベル変換にはポピュラーな1488/1489A を使用してあります．また，ドライバ側に必要な±12V の電源はカードエッジ・コネクタを通して外部から供給します．

●並列インターフェース

意外に思われるかもしれませんが，この Z80 PIO を使った並列インターフェースはほとんど使われることがありません．しいて使われる場合を掲げると，基板上の DIP スイッチの状態を読む必要のある場合と，Z80 PIO を通して割込みをかける場合くらいなものでしょう．この部分があまり使われないのは，次のような理由からです．

(1)　Z80 PIO の出力は，そのままでは長いケーブルを駆動できない．

(2)　必ず初期設定を必要とする．これは手間がかかる以外に安全性（入出力を間違えたりした場合に危険）の面で問題がある．

(3)　シーケンス・コントローラやプロセス・コントローラなどの用途では多数のディジタル入出力を必要とする．このためディジタル入出力専用の基板（後述）を組み合わせて使うので，CPU 基板で入出力を行う必要がない．

したがって，この部分は全く省略してもとくに差し支えはありません．ただし，このシリーズでは CPU 基板以外の基板は原則としてメモリ・マップトになっており，Z80 CPU の割込み機能をフルに使うには不便です．そのため，この Z80 PIO を割込みコントローラ的に使用すると便利なことがあります．

●その他

マザー・ボードに差し込み他の基板と組み合わせて使うため，データ・バスおよびアドレス・バスはバッファを介してカードエッジ・コネクタに接続されています（図6.4）．

前述のように，このシリーズは標準規格等に合わせたものではなく，独自の接続となっています．なお，基板寸法は114.3mm×203.2mm です（写真6.1参照）．

6.2　多数のスイッチの入力インターフェース

制御装置などへの応用では，パネルに取り付けられた多数のスイッチを読み込まなければならないことがあります．一口にスイッチといっても，トグル・スイッチ，押しボタン・スイッチ，ロータリ・スイッチ，ディジタル（サム・ホィール）スイッチなど多くの種類があります．しかし中でもロータリ・スイッチは1個のスイッチで多数の接点（1回転あ

図 6.4 バス・バッファ

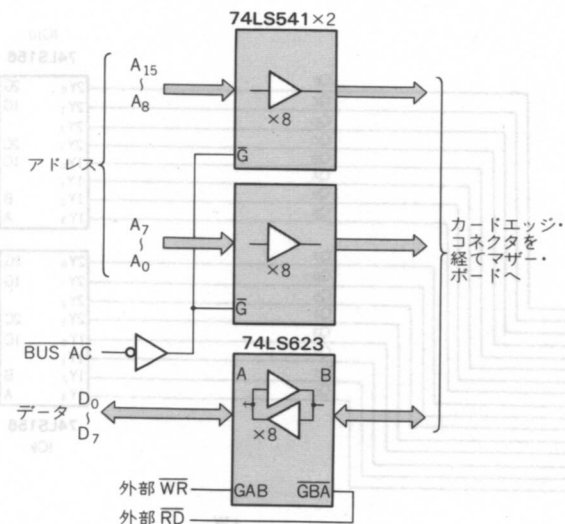
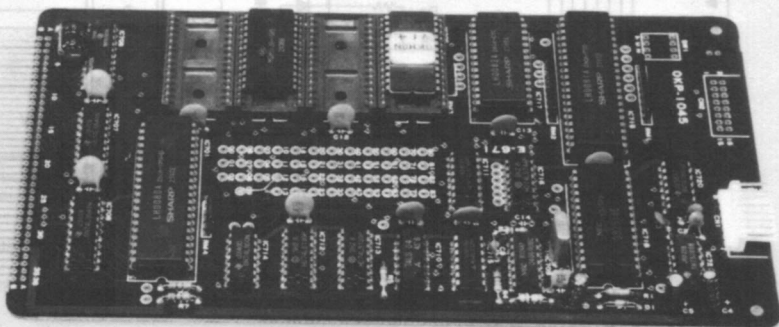
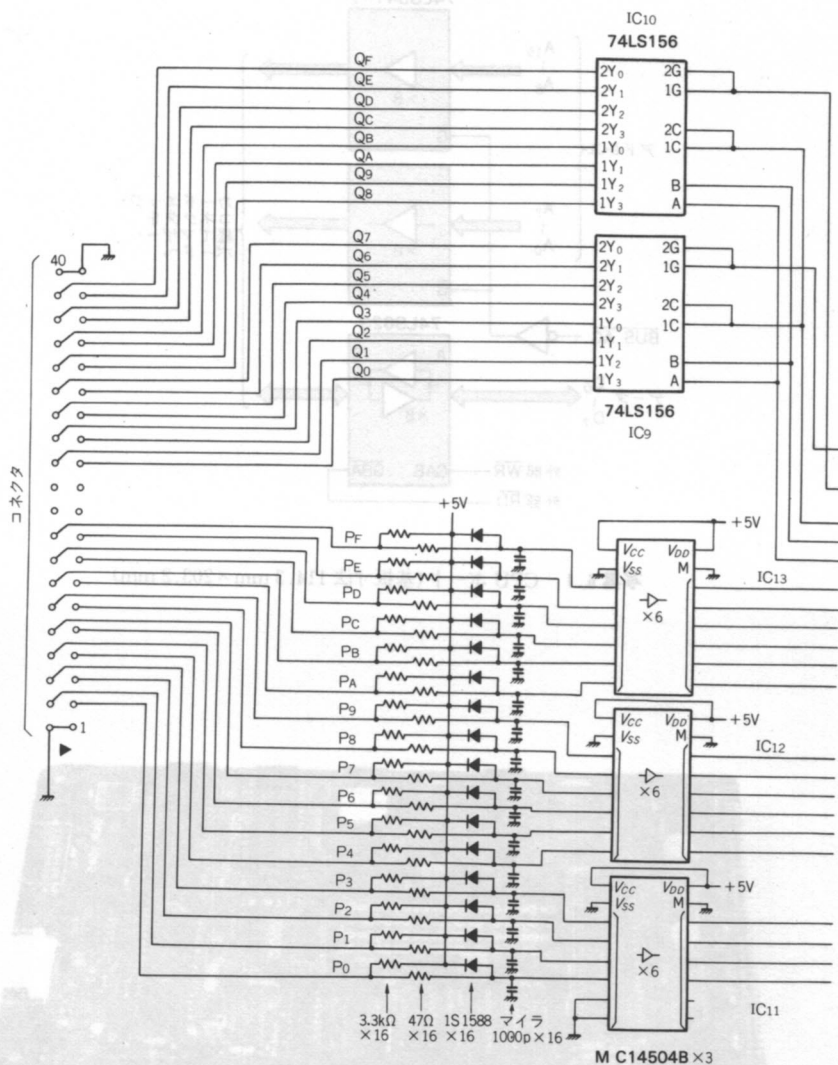
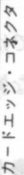



写真 6.1 CPU ボード(基板寸法 114.3 mm×203.2 mm)

図6.5　スイッチ入力インターフェース

+5V
IC3
74LS688
+5V
10kΩ×12
+5V
G
Q P
P=Q
IC2
74LS688
G
Q P
P=Q
J15
J14
J13
J12
J11
J10
J9
J8
J7
J6
J5
J4
IC5
2/4 74LS00
B
A
+5V
16V
47μ
XRD
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
カードエッジ・コネクタ
6/4 74LS04
IC6
IC8
74LS148
7
6
5
4
3
2
1
0
EI
GS
EO
A2
A1
A0
74LS148
EI
GS
IC7
4/4 74LS00
IC4
G G
×8
D7
D6
D5
D4
D3
D2
D1
D0
74LS541
IC1

たり12〜24接点)を持ち，パネル面積がかぎられている場合にセレクタとして使用するには最適です．

ところが，これを読み込む側は大変で，16接点のスイッチが16個も並べば，それだけで272本の配線が必要となります(各1本の共通点をお忘れなく)．また，これを256ビットとして読み込んでソフトウェアで解読していたのでは，ソフトの負担も大きくなります．

図6.5に示すのは，このような場合に使用するスイッチ入力インターフェースです．しかしパネルに取り付けられるのはロータリ・スイッチだけではないので，個別のスイッチ(トグル・スイッチや押しボタン・スイッチ)，あるいは2進符号出力のディジタル(サム・ホィール)スイッチなども一緒に読み込めるよう考慮してあります．

一方，長いケーブルを使って電源系統の異なる機器に接続する場合とは違い，パネルまでは数十cm程度でとくに強力な雑音源も近くに存在しないと考えられますから，一般的な雑音対策を施すだけとし，絶縁等は考慮しないことにします．

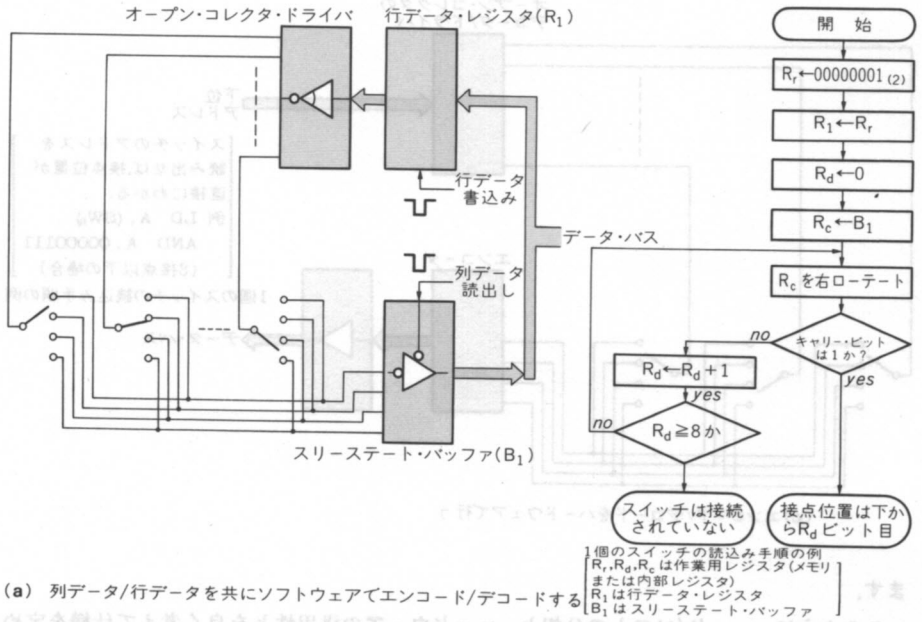
●ダイナミック・スキャン

第3章，例3-7でも説明しましたが，このような場合はダイナミック・スキャンを利用する方法があります．しかし，ダイナミック・スキャンにも様々な種類があります．図6.6(a)に示すのは，まずソフトウェアで行データを R_1 に書き込み，次に読み出した列データからスイッチの接点位置をさがすもので，ハードウェアは簡単ですが，ソフトウェアの負担は大きくなります．

図6.6(b)に示すのはソフトウェアでの行データ書込みが不要のものです．つまり，読出し動作のアドレスに応じて，自動的に1本の行をアクティブにします．

さらに，図6.6(c)に示すものでは，読み出した列データのエンコードまでハードウェアで行います．多くの場合，多接点のスイッチの接点位置は“数値”として扱うか，“テーブルを参照する”のに使うかのいずれかです．したがって，接点位置はエンコードして数値としておいたほうがソフトウェアの負担が減ります．

ただし，特定のビット(接点位置)が“1”か“0”かだけを調べたいこともあり，また同じハードウェアを個別スイッチ用にも兼用できるよう，図6.5の回路では列データ(16本)をエンコードした信号(4ビット)，スイッチが接続されているか否かを示す信号(1ビット)，列データの一部のビット・パターンをそのまま出力する信号(3ビット)をCPUから読み出せます．これで，16接点のロータリ・スイッチ16個(配線引出し本数32本)，または個別スイッチ48個(同19本)，あるいは各種スイッチの混合使用が可能となってい

図 6.6 ダイナミック・スキャンの方法

(a) 列データ/行データを共にソフトウェアでエンコード/デコードする

オープン・コレクタの
デコーダ/ドライバ

下位
アドレス

[スイッチの読込み手順は(a)に準じる.
ただし,B_1読込みの際の下位アドレス
をスイッチ番号とする.R_rは不要.]

データ
読出し

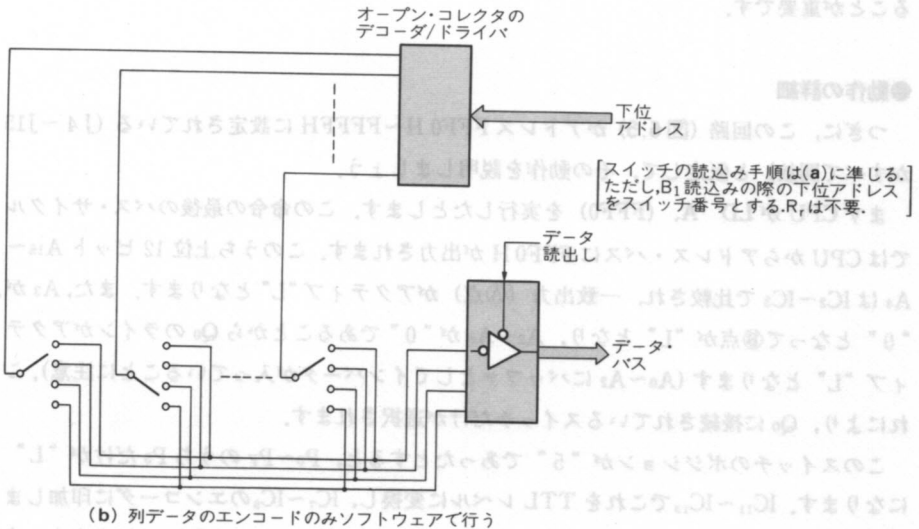
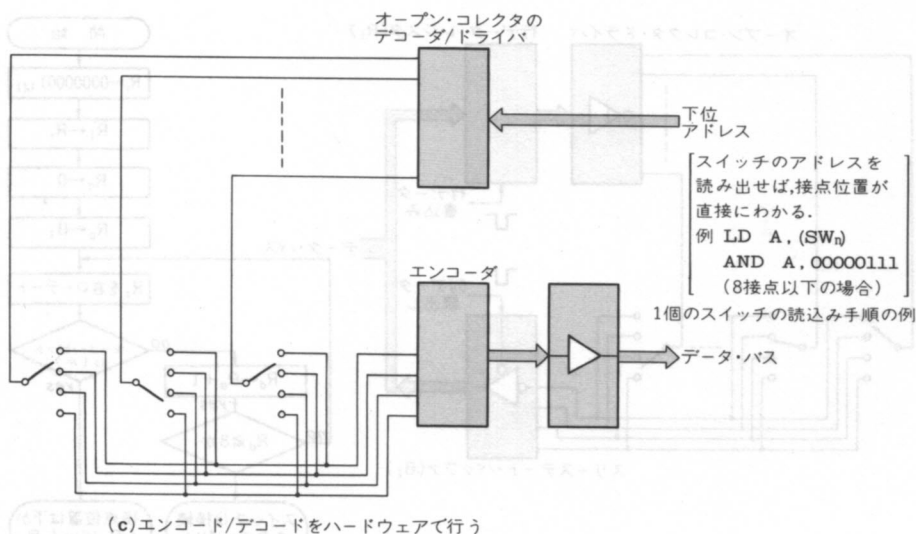
データ・
バス

(b) 列データのエンコードのみソフトウェアで行う

図 6.6 (つづき)

(c)エンコード/デコードをハードウェアで行う

ます．

このように，ハード/ソフトの分担と，ハードウェアの汎用性とを良く考えて仕様を定めることが重要です．

●動作の詳細

つぎに，この回路 (図 6.5) がアドレス FFF0 H～FFFFH に設定されている (J 4 ～J15 がすべて開放) と仮定して，その動作を説明しましょう．

まず CPU が LD　A,　(FFF0) を実行したとします．この命令の最後のバス・サイクルでは CPU からアドレス・バスに FFF0 H が出力されます．このうち上位 12 ビット A_{15}～A_4 は IC_2～IC_3 で比較され，一致出力 (Ⓐ点) がアクティブ“L”となります．また，A_3 が，“0”となってⒷ点が“L”となり，A_2～A_0 が“0”であることから Q_0 のラインがアクティブ“L”となります (A_0～A_2 にバッファとしてインバータが入っていることに注意)．これにより，Q_0 に接続されているスイッチだけが選択されます．

このスイッチのポジションが“5”であったとすると，P_0～P_F のうち P_5 だけが“L”になります．IC_{11}～IC_{13} でこれを TTL レベルに変換し，IC_7～IC_8 のエンコーダに印加します．エンコーダはこの結果を 2 進変換し，バス・インターフェース (IC_1) に加えます．少

し遅れて$\overline{\mathrm{MEM \cdot RD}}$信号がアクティブになり，$IC_7$からデータ・バスの下位4ビットにスイッチの接点位置“5”が出力されます．

このとき，D_4～D_6 には入力 P_0～P_2 の状態がエンコーダを通さないで接続されていますから，1個のロータリ・スイッチのかわりに3個の個別スイッチを接続できます（図6.7）．また，D_7 にはエンコーダから接続の有無を示す信号が接続されていて，接点数の拡張にも使えます（図6.8）．

図 6.7 スイッチの接続

Q_F ～ Q_0 のいずれか

読出し時のビット・マップ

D_7 D_0

個別スイッチのオン/オフがわかる．

ロータリ・スイッチの接点位置が2進4桁で出力される．

1つ以上の接点が閉じていたら“H”

P_F P_E P_D P_C P_B P_A P_9 P_8 P_7 P_6 P_5 P_4 P_3 P_2 P_1 P_0

ロータリ・スイッチまたはディジタル・スイッチ（10進出力）

個別スイッチ（出力線Q_0～Q_Fの1本あたり3個まで）

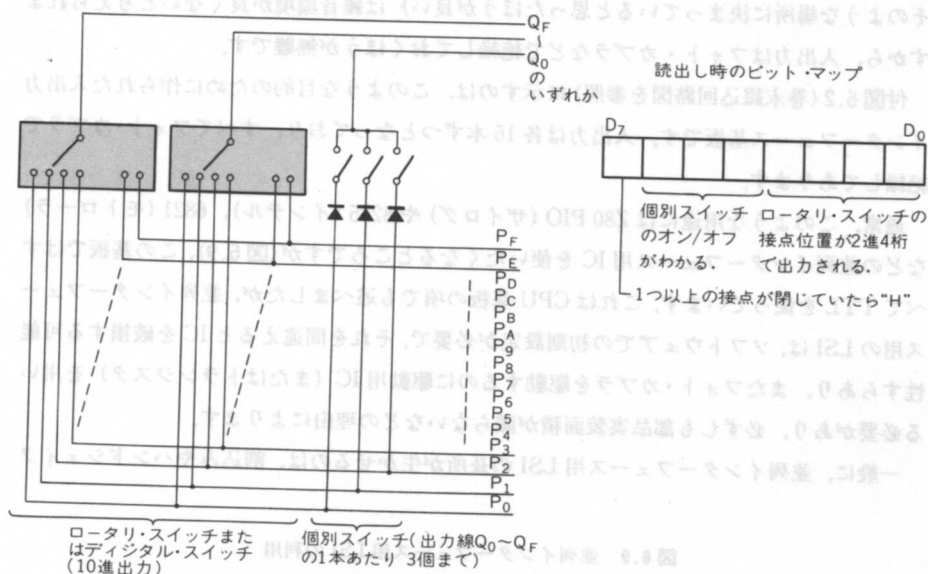
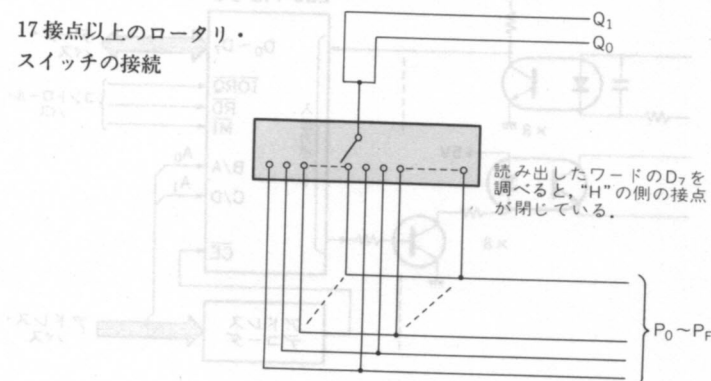



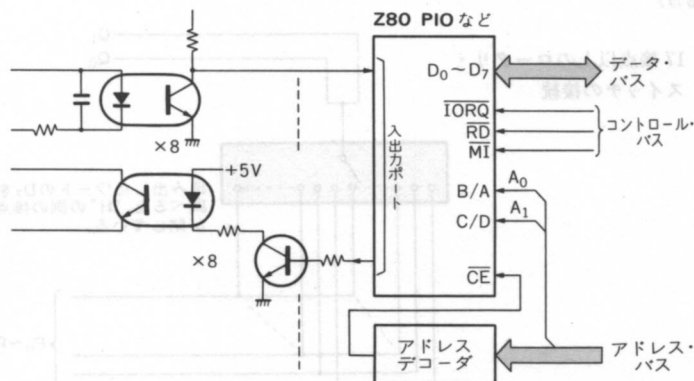
図 6.8 17接点以上のロータリ・スイッチの接続

6.3　フォト・カプラで絶縁された汎用入出力インターフェース

シーケンス・コントローラ的な用途では，多数のディジタル入出力が必要とされます．多くの場合，この信号は接点またはオープン・コレクタの形で出し入れされます．これはレベル変換が容易であるとともに，電源系統の異なる装置間での干渉を防ぐためです．また，一般にモータやソレノイドを使う場所（シーケンス・コントローラが置かれるのは，そのような場所に決まっていると思ったほうが良い）は雑音環境が良くないと考えられますから，入出力はフォト・カプラなどで絶縁しておくほうが無難です．

付図6.2(巻末綴込回路図を参照)に示すのは，このような目的のために作られた入出力インターフェース基板です．入出力は各16本ずつとなっており，すべてフォト･カプラで絶縁してあります．

通常，このような用途にはZ80 PIO (ザイログ) や8255 (インテル)，6821 (モトローラ) などの並列インターフェース用ICを使いたくなるところですが(図6.9)，この基板ではすべてTTLを使っています．これはCPU基板の項でも述べましたが，並列インターフェース用のLSIは，ソフトウェアでの初期設定が必要で，それを間違えるとICを破損する可能性すらあり，またフォト･カプラを駆動するのに駆動用IC（またはトランジスタ）を用いる必要があり，必ずしも部品実装面積が減らないなどの理由によります．

一般に，並列インターフェース用LSIの長所が生かせるのは，割込みやハンドシェイク

図6.9　並列インターフェース用LSIの利用

機能を活用する場合にかぎられます.

●アドレス・デコーダ

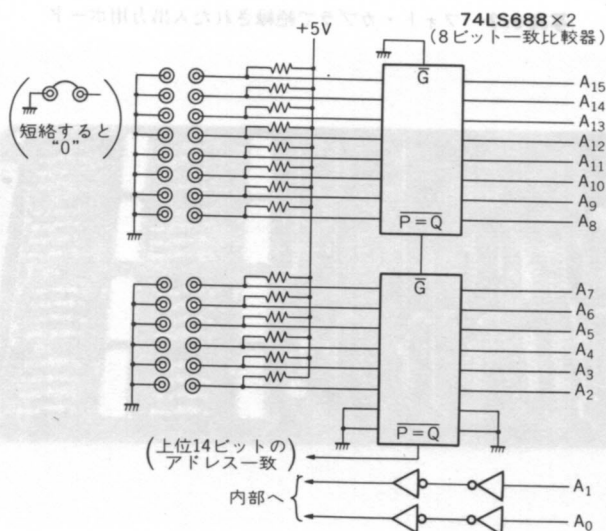
この基板の入出力は各16本（2バイト）ずつで，出力側のデータの読み返しができるので4バイトのアドレスが必要です.用途によっては，ディジタル入出力の本数が16本ずつでは足りないこともあり，その場合はこの基板を2枚以上使いますが，その場合にもソフトウェアから見て連続したアドレスになったほうが何かと便利です.

以上の点から，この基板は上位14ビットのアドレスのフル・デコードができるようにしてあります.アドレス・デコーダとしては8ビットのコンパレータ（74LS688）を使っています（図6.10）.

●入出力の極性の変更など

この回路では，入出力ともに接点がオンした状態がソフトウェア上での"1"に，オフが"0"に各々対応しています（図6.11）.これを逆にするには，付図6.2の回路中のIC_9～IC_{12}を非反転の74LS253に，IC_1を同じく74LS541に変更します.

また，図の回路は入力用として絶縁された12Vの電源を仮定していますが，用途によっ

図6.10　アドレス・デコーダ

ては5Vや24Vになることもあります．その際にはコネクタ側の抵抗を変更し，フォト・カプラの1次側に10mA前後の電流が流れるようにします．

●入出力の本数と汎用性

写真6.2からおわかりのように，この基板はガラ空きです．基板寸法と回路規模とを考

図6.11 ソフトウェアから見た論理(1/0)の対応

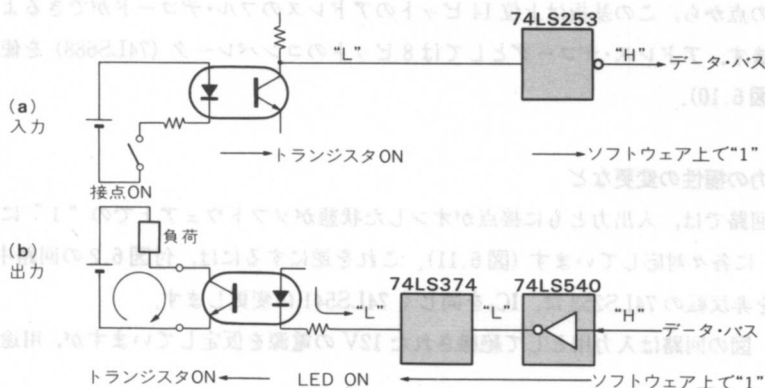



写真6.2 フォト・カプラで絶縁された入出力用ボード

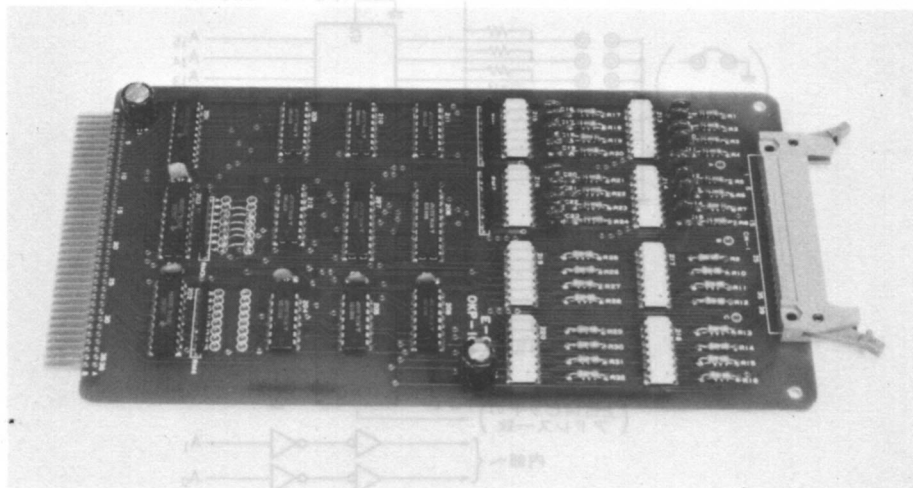
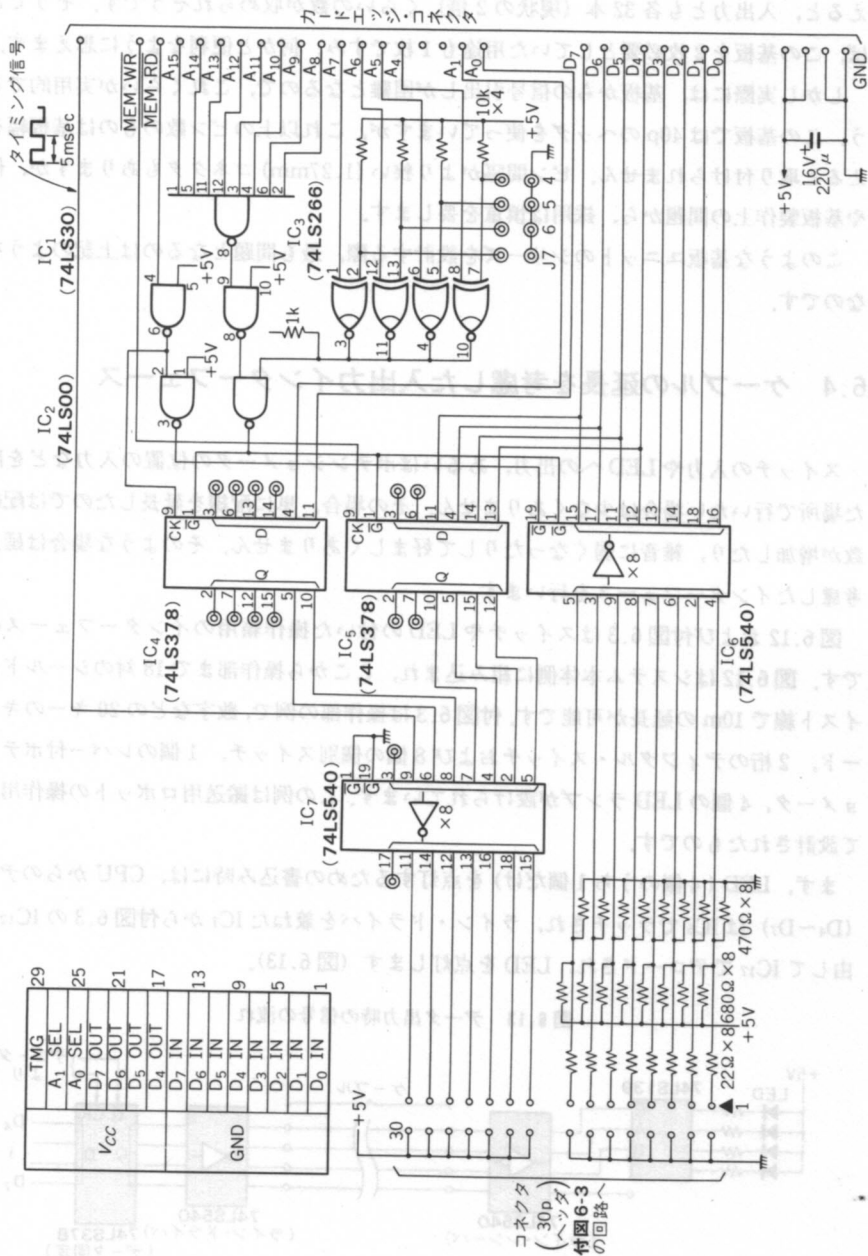



図 6.12 操作部用インターフェース

えると，入出力とも各32本（現状の2倍）くらいの数が収められそうです．そうできれば，この基板を2枚必要としていた用途も1枚ですみ，何かと便利なように思えます．

しかし実際には，基板からの信号引出しが困難となるので，これくらいが実用的でしょう．この基板では40pのヘッダを使っていますが，これ以上のピン数のものは基板幅を考えると取り付けられません．ピン間隔がより狭い(1.27mm)コネクタもありますが，価格や基板製作上の問題から，採用は慎重を要します．

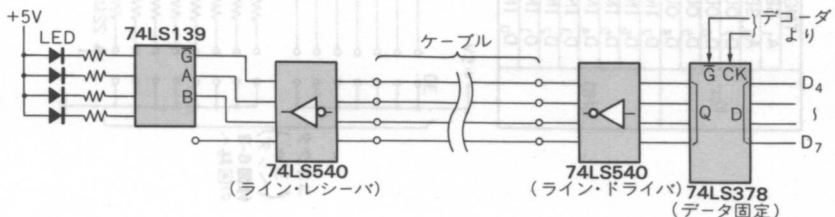
このような基板ユニットのシリーズを設計する際，最も問題となるのは上記のような点なのです．

6.4　ケーブルの延長を考慮した入出力インターフェース

スイッチの入力やLEDへの出力，あるいはポテンショメータの位置の入力などを離れた場所で行いたい場合は少なくありません．その場合，単に配線を延長したのでは配線本数が増加したり，雑音に弱くなったりして好ましくありません．そのような場合は延長を考慮したインターフェースを行います．

図6.12および付図6.3はスイッチやLEDの付いた操作箱用のインターフェースの例です．図6.12はシステム本体側に組み込まれ，ここから操作部まで18対のシールド付ツイスト線で10mの延長が可能です．付図6.3は操作部の例で，数字などの20キーのキーボード，2桁のディジタル・スイッチおよび8個の個別スイッチ，1個のレバー付ポテンショメータ，4個のLEDランプが設けられています．この例は搬送用ロボットの操作用として設計されたものです．

まず，LED(4個のうち1個だけ)を点灯するための書込み時には，CPUからのデータ(D_4～D_7)はIC_5でラッチされ，ライン・ドライバを兼ねたIC_7から付図6.3のIC_{12}を経由してIC_{17}でデコードされ，LEDを点灯します（図6.13）．

図6.13　データ出力時の信号の流れ

スイッチやポテンショメータの状態を読み出す場合には，$\overline{RD}$の前縁でアドレスの下位2ビット（A_0，A_1）をラッチし(IC_4)，その状態を操作箱側に送ります．この信号はIC_{12}を経てマルチプレクサ（IC_{13}～IC_{16}）に印加され，必要なデータを選択します．

マルチプレクサで選択されたデータはIC_{11}からライン・レシーバ兼バス・インターフェースのIC_6に送られ，データ・バスに出力されます（図6.14).

このような用途では，一般に雑音環境が良くないので，伝送距離に応じた駆動能力や終端条件を考慮する必要があります．この例ではTTLのバス・ドライバである74LS540をライン・ドライバ/レシーバとして使っていますが，距離がさらに長くなる場合は差動にするなどの方法を採ります．

次に各信号はなるべくスタティックな状態で伝送するようにします．この例ではアドレス（A_0～A_1）をラッチしてから伝送するとともに，$\overline{RD}$や$\overline{WR}$等のパルス状のストローブ信号はケーブルには乗せていません．

第3に，信号線数を減らすよう配慮することです．たとえば，直列転送を使えばソフト/ハードは複雑ですが信号線としては少なくなります．

6.5 分散処理のための直流モータ制御用基板

直流サーボ・モータの制御（速度，加速度，位置決めなど）をマイコンで行うのは結構面倒です．これは，ある程度のリアルタイム性が必要な上，数値計算を伴うためです．ところが，多軸のロボットなどで複数のモータを同時に制御しなければならないことが増してきています．そこで，分散処理を行います．

図6.14 データ入力時の信号の流れ

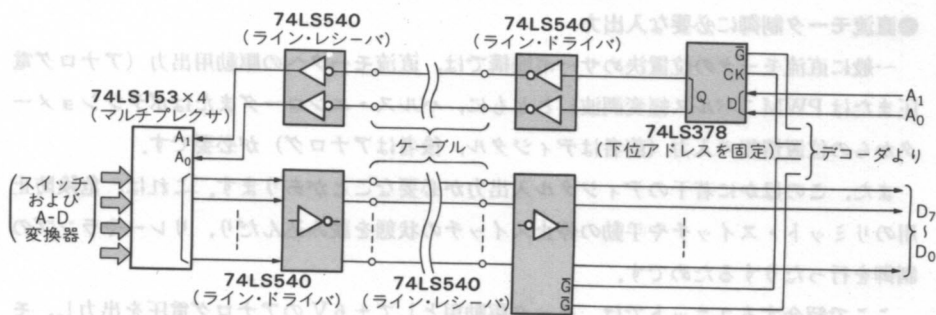
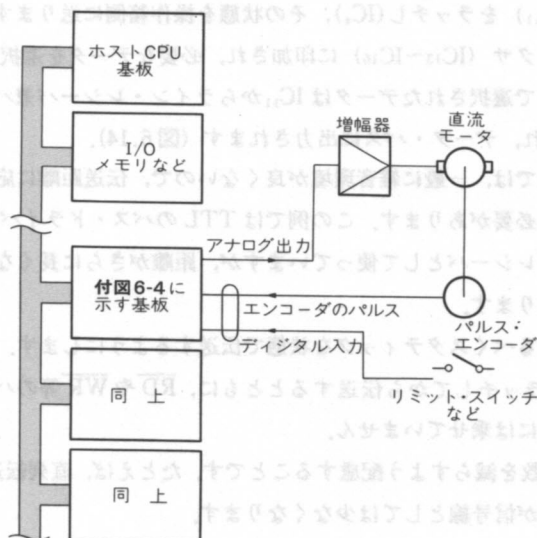



図 6.15　多軸サーボ・システムの構成例

分散処理にも様々な形態が考えられますが，ここで紹介するユニットは図 6.15 のような構成で使用します．このような構成にすると1個の CPU でモータを制御するのに比べてハードウェアが複雑になるように思えます．しかし，各モータ制御ユニットはすべて同じものであり，モータ数の増減などに対して柔軟性があります．

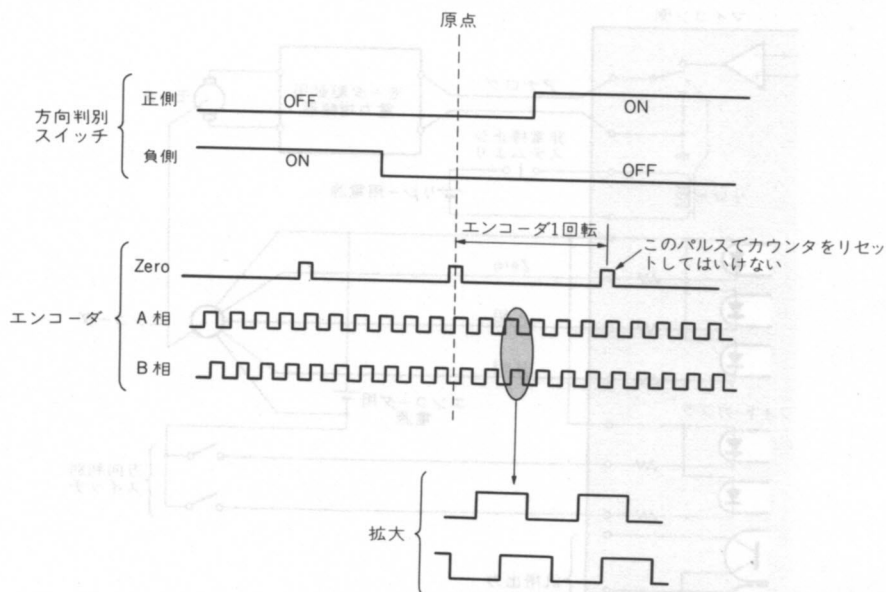
それにも増して重要なのはソフトウェア上の利点です．すなわち，各モータの特性の違いやモータ数の変化に対して，ホスト CPU のソフトウェアはほとんど関知する必要がありません．したがって，大規模であっても柔軟性に富んだシステムがつくれます．

●直流モータ制御に必要な入出力

一般に直流モータの位置決めサーボ機構では，直流モータへの駆動用出力（アナログ電圧または PWM：パルス幅変調波）とともに，パルス・エンコーダまたはポテンショメータからの位置情報の入力（前者はディジタル，後者はアナログ）が必要です．

また，このほかに若干のディジタル入出力が必要なことがあります．これは，危険防止用のリミット・スイッチや手動の停止スイッチの状態を読み込んだり，リレーやランプの制御を行ったりするためです．

ここで紹介するユニットでは，モータ駆動用として±6Vのアナログ電圧を出力し，モ

図6.16　エンコーダの出力と方向判別用スイッチとの関係

ータを実際に駆動する電力増幅器は外付けとします.

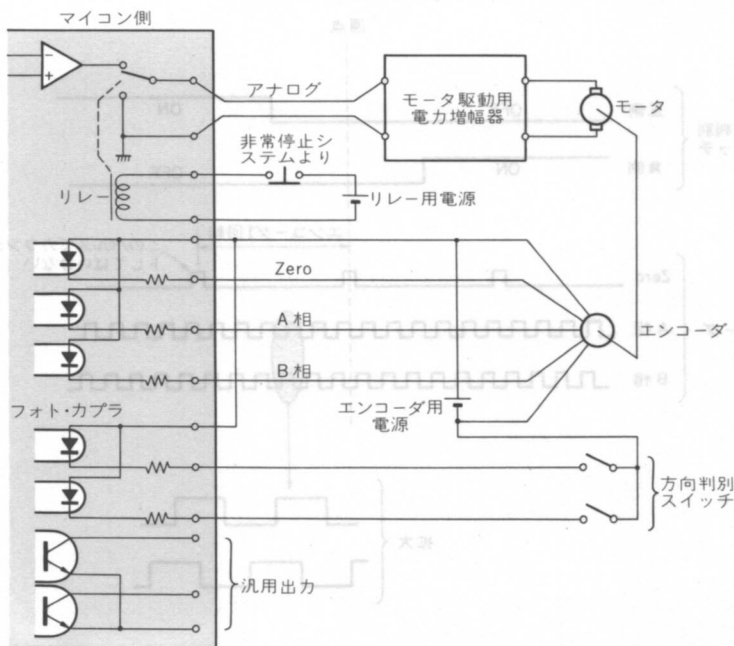
また，位置検出用としてはパルス・エンコーダを使います．パルス・エンコーダからは1回転に1個のリセット・パルスが出力されますが，これを多回転で用いる場合，カウンタのリセットを防止する必要があります．このために方向判別用の2個の接点信号を入力できるようになっています（図6.16を参照).

このほか，フォト・カプラで絶縁された2ビットの出力，外部からアナログ出力をオン/オフするためのリレーなどを備えています．後者は万一の非常停止などの際に使用します（図6.17).

● CPUの選定

この基板の回路を付図6.4に示します．一見したところ，Z80を中心とした独立したマイコン基板のように見えます（事実，単独で動かすことも可能です）が，前述のようにスレーブとして動くよう設計されたものです.

ただし，このように入出力やメモリがかぎられており拡張の必要のない用途には，Z80の

図 6.17　直流モータ制御用入出力の例

ような汎用 CPU は適切ではありません．一般には，8041A や 8048/51 系，F 8，6801，Z 8 などのシングルチップ・コンピュータを使い，部品数を削減すべきです．

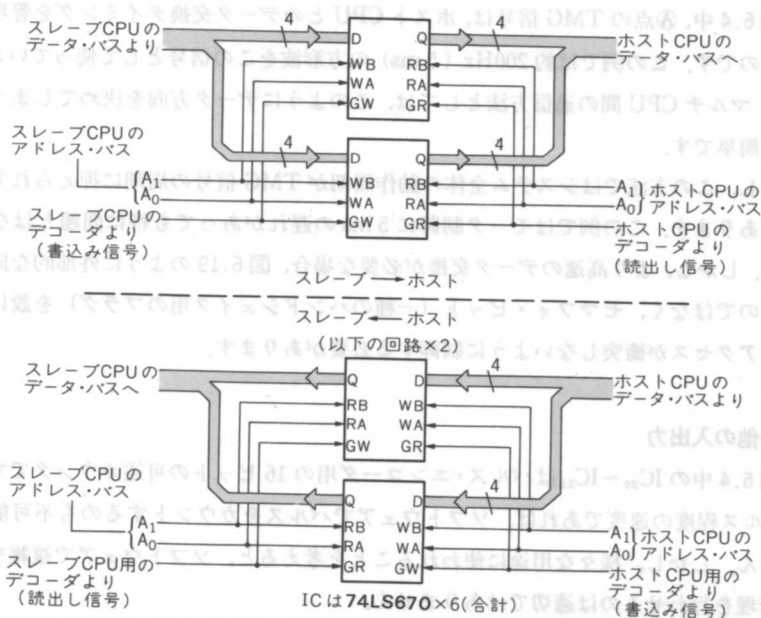
しかし，ソフトウェアの開発環境や部品管理上の問題からハードウェアの統一を要求される場合には，ホストもスレーブも Z80 といった事態が生じるのです．

●ホストとの間のデータ交換

CPU として 8041A や Z 8 を使えば，ホストは直接にスレーブ CPU の内部レジスタをアクセスできます．しかし，Z80 CPU ではそうはいきませんから，何らかのデータ交換方法を設けなければなりません．

この基板の用途ではホストからスレーブに渡すデータは 4 バイト（サーボ・モータの停止目標位置 16 ビット，速度・加速度のパラメータ各 8 ビット），スレーブからホストに渡すデータは 3 バイト程度（現在位置 16 ビットと若干のステータス・フラグ）です．

そこで，この回路では IC_{10}～IC_{15} (74LS670) を使い，ホスト→スレーブは 8 バイト，ス

図 6.18 データ交換レジスタ部

レーブ→ホストは4バイトのデータ交換レジスタ群としています(図6.18).このレジスタ群はホスト/スレーブのいずれから見てもメモリ・マップされた入出力ポートに見えます.部品数を少なくするため,出力したデータの読み戻しはできません.

データ交換用としては,このほかにFIFOを使う方法(第5章,例5-4を参照)や直列通信を用いる方法,があります.いずれも非同期的に大量のデータに受け渡しするには適していますが,少量・高速のデータ交換には向きません.

また,この基板ではデータ交換レジスタとしてTTLのレジスタである74LS670(4語×4ビット)を用いています.これは,設計時点(1983年)で適当なデュアル・ポートRAMやバス・レジスタが入手できなかったためです.74LS670は速度の点では充分ですが,1個あたりの容量が少なく消費電流が多いなどの欠点があります.現在ではこれに相当するCMOSのレジスタ,または両側から読み書きできるデュアル・ポートRAMが入手できるので検討をおすすめします.

●データ交換のタイミング

付図6.4中，Ⓐ点のTMG信号は，ホストCPUとのデータ交換タイミングを管理するためのものです．この例では約200Hz（5ms）の方形波をこの信号として使っています（図6.19）．マルチCPU間の通信方法としては，このようにデータ方向を決めてしまうのがもっとも簡単です．

ただし，この方法ではシステム全体の動作周期がTMG信号の周期に押えられてしまう欠点があります．この例ではモータ制御に5msの遅れがあっても特に問題とはなっていません．しかし，より高速のデータ交換が必要な場合，図6.19のように外部的な同期信号を使うのではなく，セマフォ・ビット（一種のハンドシェイク用のフラグ）を設けて両側からのアクセスが衝突しないように制御する必要があります．

●その他の入出力

付図6.4中のIC_{29}～IC_{32}はパルス・エンコーダ用の16ビットの可逆カウンタです．毎秒100パルス程度の速度であれば，ソフトウェアでパルスをカウントするのも不可能ではありません．しかし，様々な用途に使われることを考えると，ソフトウェアで複雑なタイミング管理を行わせるのは適切ではありません．

ここで使用するようなインクリメント型のエンコーダでは，電源を投入してから原点を通過するまでは正確な位置がわかりません．そこで，IC_{28}から成るRSフリップフロップを使い，原点を通過したか否かを覚えておきます．

IC_2，IC_3は12ビットのD-A変換器です（図6.20）．ここでは上位8ビットをデータ・バス，下位4ビットをアドレス・バスに各々接続し，一度に12ビットの書込みができるよう

図6.19　CPU間のデータ交換タイミング

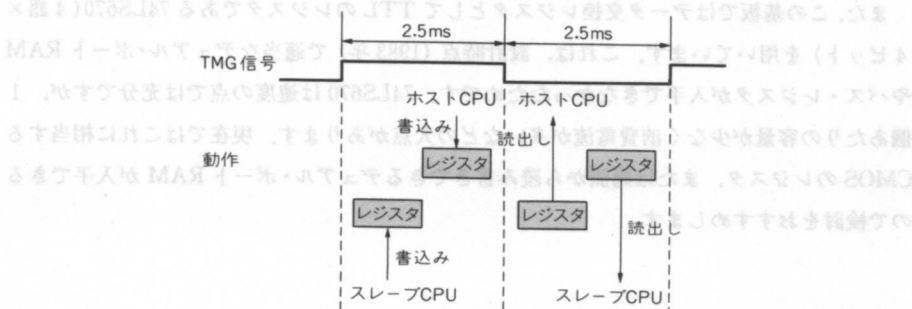



図 6.20 D–A 変換部

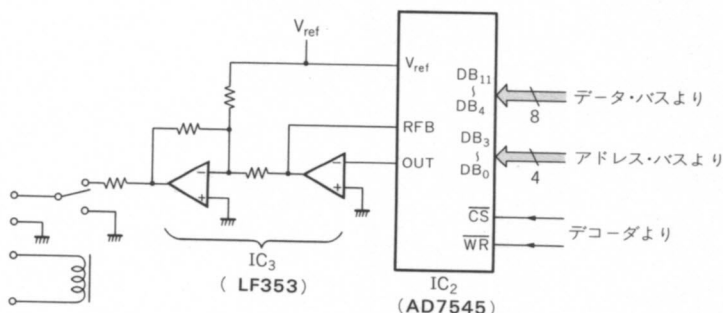
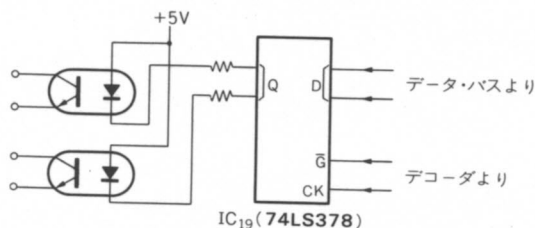



図 6.21 ディジタル出力部

になっています.

IC_3からのアナログ出力はリレー接点を介して基板外に導かれます.ここに入っているリレーは，電源投入時などアナログ出力が不定の期間にアナログ電圧をモータに印加しないよう遮断したり，非常停止の際の制御に使ったりするためのものです.

このほか，若干のディジタル入出力がありますが，基板外に引き出す可能性のあるものはすべてフォト・カプラで絶縁してあります．一般に，近くで複数のモータが回っているような場所は，雑音環境がかなり悪いと考えられます．また，ディジタル入出力の相手側はリミット・スイッチやシーケンサなど，何をつながれるかわかりません．そこで，最も無難な方法としてフォト・カプラで絶縁しておきます.

ディジタル入力のほうは，エンコーダ用カウンタや基板上のDIPスイッチなどとともにIC_{20}～IC_{23}（マルチプレクサ）を使ってデータ・バスに接続してあります（図6.21)．一方，ディジタル出力はIC_{19}（Dフリップフロップ）から取り出しています．このへんは第5章で述べた通りの基本的なものです.

Appendix

マイコン・システムのデバッグの例

── Z80用モニタ・プログラム ──

I．バグをなくすために

万一（残念ながら万に数千の確率で起こるのですが）作った装置が思い通りの動作をしなかった場合，その原因を追求しなければなりません．その装置がマイコンを含んでいる場合には，ソフト/ハードのどちらに原因があるかわからないと手の下しようがありません．これがマイコン・システムのデバッグの面倒さにつながっています．

では，どのようにすればバグの少ない（あるいは見つけやすい）システムになるかを考えてみます．

●設計の注意点

極力簡単なハードウェアを設計すべきです．ただしこれは単に部品数が少ないことを意味するのではありません．たとえば簡単な入出力ポートをTTLやCMOSのMSIで作った場合と，並列インターフェース用のLSI（Z80 PIOや8255など）を使った場合とを比較すると，前者のほうが一見複雑に思われます．

しかし，後者はソフトウェアでの初期設定をしないと動かせないことなどを考えると，必ずしもバグが取りやすいとはいえません．

●製作上の注意点

手作りの装置に見られる欠陥は，ハード/ソフトともに大部分が製作時の人為的ミスといって良いでしょう．それも，ハンダ付けの不良とか，ジャンプ先の間違いなど，いたって初歩的なものが多いようです．

このようなミスは絶対に起こさないよう努めるのが第1です．しかし，それだけでは不充分で，起こしたミスを発見しやすいような設計を行っておくことも必要です．これには，次のような方法が考えられます．

(1)　適切なブロック化，モジュール化を行う．ソフトウェアでは使用言語によらず50～60行以下，ハードウェアではIC 20～30個以下を1ブロックとすれば，検査が比較的容易である．

(2)　製作ルールを明確にする．ソフトウェアではラベルや変数名の統一をはかることなど．ハードウェアでは配線の手順や色分けなど．

(3)　間違いは必ずあると思って良い．系統的な検査をすること．

II．Z80用モニタ・プログラム

“作った装置が動かない．しかし原因がハードかソフトかもわからない”といった事態を避けるには，間違いのないデバッグ用ソフトウェア（簡単なモニタ・プログラム）を用意

図1　モニタ・プログラムを使うためのハード構成

(a)　V.1.4用

(b)　V.1.2用

表A　実行できるコマンド例

コマンド	説明
G [〈Add〉] ↵	〈Add〉番地へ実行を移す．〈Add〉が省略された場合は0とみなされる．〈Add〉を4桁以上入力したときは最後の4桁が有効となる．
R ↵	RST 38Hでブレークしたアドレスにリターンする．表のレジスタは復改される．ユーザ・ルーチン内でRST 38Hでブレークした時には，その命令をもとにもどしてからリターンしなければならない．
M [〈SAdd〉], [〈EAdd〉], [〈TAdd〉] ↵	〈SAdd〉から〈EAdd〉までのメモリの内容を〈TAdd〉からに移動する．
D [〈SAdd〉], [〈EAdd〉] ↵	〈SAdd〉から〈EAdd〉までのメモリの内容を表示する．〈SAdd〉を省略するとその前に使用したアドレスの値になり，〈EAdd〉を省略すると，〈SAdd〉+3FHとなる．
S [〈Add〉] ↵	〈Add〉を入力すると，その番地の内容を表示して止まるので変更する時はその値2バイトを入力し，変えない時はスペースを入力する．アドレスをもどすときはBack Spaceを入力する．〈Add〉を省略するとその前に使用したアドレスの値になる．
F 〈SAdd〉, 〈EAdd〉, 〈CONST〉	〈SAdd〉から〈EAdd〉までの内容を〈CONST〉でうめる．各アドレスとデータは省略できない．
X	‘X’を入力するとブレークした時のPC, AF, BC, DE, HL, SPの内容を表示する．
I [〈Add〉] ↵ I [〈Add〉],	〈Add〉ポート番地の内容を表示する．省略した場合は0となる．‘↵’の場合は次の行に，‘,’の場合はその行に内容を表示する．
O [〈Add〉], 〈Data〉 ↵	〈Add〉ポート番地に〈Data〉を出力する．

しておくことです．ただし，ソフトウェアもハードの構成によって変更する必要があります．したがって，少なくとも一部の入出力のアドレス配置は統一しておくと便利です．

ここで紹介するモニタ・プログラムは図I(a)または(b)のハードウェアを前提としています．このうち(a)のほうは第6章のCPU基板で使用できます．また，(b)のほうはCTCや8251Aを持たないシステムであっても，I/O領域に入出力ポート（Z80 PIOまたはハードウェアによるもの）があれば使用できます．ただし，いずれもクロック周波数とボーレートとが関係するので，クロック周波数を変更する際は若干の修正が必要です．

このモニタ・プログラムで実行できるコマンドは表Aの通りです．これは日本電気のPC 8801の内蔵のモニタ・プログラム，あるいはCP/M上で走るSID（シンボリック・デバッガ）と良く似ていますので，これらを開発装置として使う際にも便利です．

なお，この程度の簡単なモニタ・プログラム（198ページ参照）は，使用するCPUを理解するのに最適です．したがって，初めて使用するCPUには自分でモニタ・プログラムを作ってみると，ハード/ソフトともに参考となるでしょう．

III．デバッグの実際

作った装置をデバッグするには，前述のモニタ・プログラムを書き込んだROMを実装し，充分な検査の終ったハードウェアにターミナル（またはパーソナル・コンピュータをターミナル・モードとしたもの）を接続し，電源を投入すれば良いわけです．するとターミナルの画面に，

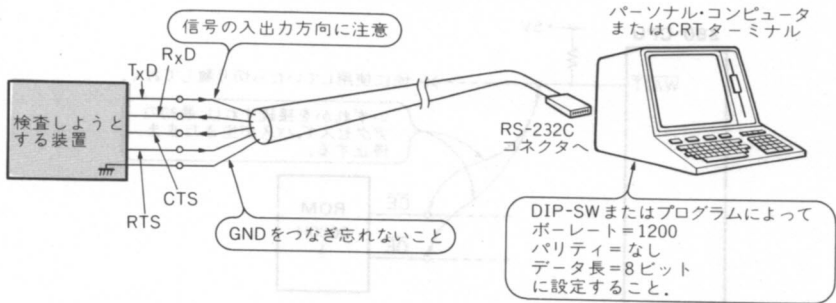
```
  RAM  AREA  1000  1FFF
*
```

と表示が出るでしょう．出た場合は次項を飛ばして197ページをご覧下さい．しかし万一(たぶん万に数千の率ですから悲観しないように……）出ない場合は，以下を良くお読み下さい．

●ハードウェアのデバッグ

モニタ・プログラムが走らない場合，ハードウェアに何らかの異常があると考えられます．ただし，その前に次の点を確認して下さい．

(1) ターミナル(またはパーソナル・コンピュータ)との直列通信回線の設定に間違いは

図II　直列通信回線の接続

ないか？たとえばRS-232Cコネクタの入出力の違い，レベル（±12VかTTLレベルか）はどうか，パリティやストップ・ビット，信号の正負の違い，CTSやRTSのレベルの違い，ボーレート（前述のモニタ・プログラムは1200BPS用）の違いなど（図IIを参照）．

⑵　プロンプト（＊）が連続して出続ける場合はターミナルからスペースか CR （キャリッジ・リターン）を送ってみる．

⑶　何も出ない場合，作った装置のCPUに念のためリセットをかけてみる．

以上の3点で解決すれば，ハードウェアの異常ではありませんので以下を飛ばして196ページからご覧下さい．これで解決しない場合は，いよいよハードウェアの検査が必要になります．

ハードウェアの本格的な検査には2現象以上のオシロスコープが必要です．ただし，場合によってはテスタだけで見当をつけなければならないこともあるでしょうから，そのような場合も考慮して話を進めます．

ハードウェアの検査といっても様々な方法があるでしょうが，たとえば次のような手順で行います．

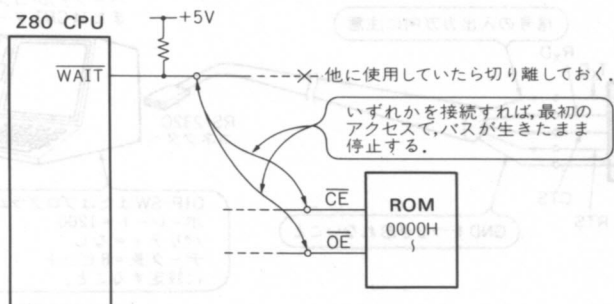
⑴　各ICには所定の電源が供給されているか？

⑵　クロック発振が行われているか？

⑶　CPUは動いているか？

これはZ80 CPUの$\overline{\mathrm{M1}}$出力をオシロスコープで見るのが簡単です．

以上がOKでしたらハードウェアの致命傷はありません．次にもう少し本格的な検査を

図III　CPUの$\overline{\text{WAIT}}$入力を使う方法

図IV　モニタ・プログラムの使用例(下線部を使用者が打ち込む. ↵はCR)

```
*SFF00↵
FF00  FF-00       ……FF00番地への書込みデータ
?                 ……読み戻せないと？が出る
*S↵               ……次回はアドレス不要
FF00  FF-         ……以下, 繰り返して必要なデータを書き込み, 確認する
```

(a)　メモリ領域 0FF00番地の出力ポートの検査

```
*I80,20           ……番地のあとに,を打つと改行なしにデータを読み込む
*I80,21           ……入力状態を変えて必要な回数の検査を繰り返す
*I80,22
```

(b)　I/O領域 80番地の入力ポートの検査

行います.

(4)　CPUの$\overline{\text{HALT}}$出力を調べる. 前述のモニタ・プログラムでは, RAMが実装されていないと判断するとHALTとなります. したがって, 電源投入後, 1秒かそこら(リセット回路等により異なる)で$\overline{\text{HALT}}$が“L”になるようなら, RAMの非実装か配線の異常です. また, モニタ・プログラム自体が非実装だとCPUがFFH (RST 38H)を2回実行してHALTとなることがありますから, ROMの配線ミスも調べます.

(5)　ROMの$\overline{\text{OE}}$をCPUの$\overline{\text{WAIT}}$入力に接続してバスの状態を調べます. 上記(4)まで異常が見られないのなら, バスの異常(ショートや開放など)を調べます. 図IIIに示すようにCPUの$\overline{\text{WAIT}}$入力を使う方法は, テスタ1台しか手許にない場合にも非常に有効で簡単に実行できるので, 是非覚えておいて下さい.

たとえば, ROMの$\overline{\text{OE}}$を接続すれば最初にROMを読み出した瞬間にWAITがかかり, アドレス・バスは0000H, データ・バスはその内容になるはずです. 同様にして他

のI/O用のICやメモリの$\overline{CE}$, $\overline{OE}$, $\overline{WE}$, あるいはアドレス線をうまく$\overline{WAIT}$とつなげば，各バスの状態がスタティックで（したがってテスタ1台でも）観察でき，非常に役に立ちます.

設計が正しく，部品に異常がなく，しかも配線に大きなミスがなければ，以上の検査で大半のミスが見つかるものです．ここまででミスが発見できない場合，あせらずにもう一度，配線の確認をおすすめします.

配線のミスは慣れた人でも1%～数%ありますが，部品の不良はそれより1桁以上少ないのです.

●モニタ・プログラムによる各部の検査

いよいよモニタ・プログラムが動いたら，これを使って各部の検査を行います．たとえば，メモリ領域への書込み・読出しにはSコマンドを使います．この場合，書き込んだ内容が同じアドレスで読み出せないと，?を表示しますが異常ではありません.

また，I/O領域はIコマンドおよびOコマンドを用いますが，こちらは同じアドレスで読み返せなくても構いません.

この様子を図IVに示すので参考として下さい.

〈リスト〉
Z80用モニタ・プログラム

```
                                ;
                                ;       OKmon  Ver 1.2 (PIO Version)
                                ;
                                ;       OK monitor Version 1.2
                                ;       PIO Version
                                ;               1983.03.08
                                ;
0000'                                   ASEG
                                        .Z80

0000                            PIO     EQU     0
                                ;
                                        ORG     0
0000    F3                      ENTRY:  DI
0001    3E CF                           LD      A,0CFH
0003    D3 02                           OUT     (PIO+2),A       ;PIO MODE 3 SET
0005    3E BE                           LD      A,0BEH
0007    D3 02                           OUT     (PIO+2),A
0009    3E 03                           LD      A,03H
000B    D3 02                           OUT     (PIO+2),A       ;PIO DI
000D    3E FF                           LD      A,0FFH
000F    D3 00                           OUT     (PIO),A
0011    C3 0352                         JP      MSRCH
                                ;
                                        ORG     30H
0030    E5                      COMP:   PUSH    HL
0031    A7                              AND     A
0032    ED 52                           SBC     HL,DE
0034    E1                              POP     HL
0035    C9                              RET
                                ;
                                        ORG     38H
0038    E3                      RST38:  EX      (SP),HL         ;RST 38H ENTRY
0039    2B                              DEC     HL              ;RETURN ADDRESS DEC.
003A    E3                              EX      (SP),HL
003B    F5                              PUSH    AF              ;SAVE REG.
003C    C5                              PUSH    BC
003D    D5                              PUSH    DE
003E    E5                              PUSH    HL
003F    21 000A                         LD      HL,0AH
0042    39                              ADD     HL,SP
0043    E5                              PUSH    HL              ;SAVE SP
0044    2B                              DEC     HL
0045    56                              LD      D,(HL)
0046    2B                              DEC     HL
0047    5E                              LD      E,(HL)
0048    EB                              EX      DE,HL           ;BREAK ADDRESS->HL
0049    CD 030A                 MON:    CALL    LINOUT
004C    0D 0A                           DEFB    0DH,0AH
004E    2A                              DEFB    '*'
004F    FF                              DEFB    0FFH
0050    CD 012F                         CALL    SEIN
                                ;
0053    FE 47                           CP      'G'
0055    20 1B                           JR      NZ,RTN
                                ;
0057    CD 01BE                 GUSR:   CALL    GETAD           ;GO USER ROUTINE
005A    20 2A                           JR      NZ,ERR
005C    CD 00BB                         CALL    CRLF
005F    EB                              EX      DE,HL
0060    E1                              POP     HL              ;RECOVER SP
0061    F9                              LD      SP,HL
0062    EB                              EX      DE,HL
0063    E9                              JP      (HL)
                                ;
                                        ORG     66H
0066    D9                      NMI:    EXX                     ;NMI ENTRY
0067    D1                              POP     DE
```

```
0068    21 006E                 LD      HL,NMI1
006B    E5                      PUSH    HL
006C    ED 45                   RETN                    ;IFF2->IFF1
006E    D5              NMI1:   PUSH    DE
006F    D9                      EXX
0070    18 C6                   JR      RST38
                        ;
0072    FE 52           RTN:    CP      'R'
0074    20 19                   JR      NZ,MON1
                        ;
0076    CD 012F                 CALL    SEIN            ;RETURN FROM RST38
0079    FE 0D                   CP      0DH
007B    20 09                   JR      NZ,ERR
007D    CD 00BB                 CALL    CRLF
0080    E1                      POP     HL
0081    E1                      POP     HL              ;RECOVER REG.
0082    D1                      POP     DE
0083    C1                      POP     BC
0084    F1                      POP     AF
0085    C9                      RET
                        ;
0086    CD 030A         ERR:    CALL    LINOUT          ;ERROR
0089    0D 0A                   DEFB    0DH,0AH
008B    3F                      DEFB    '?'
008C    FF                      DEFB    0FFH
008D    18 BA                   JR      MON
                        ;
008F    FE 4D           MON1:   CP      'M'
0091    CA 01DF                 JP      Z,MEM           ;MOVE MEMORY
0094    FE 44                   CP      'D'
0096    CA 0212                 JP      Z,DMP           ;DUMP MEMORY
0099    FE 53                   CP      'S'
009B    CA 025B                 JP      Z,SETM          ;SET MEMORY
009E    FE 46                   CP      'F'
00A0    CA 02BE                 JP      Z,FILM          ;FILL MEMORY WITH CONSTANT
00A3    FE 58                   CP      'X'
00A5    CA 0316                 JP      Z,XMINR         ;EXAMINE REGISTER
00A8    FE 49                   CP      'I'
00AA    CA 02E0                 JP      Z,INP           ;INPUT PORT
00AD    FE 4F                   CP      'O'
00AF    CA 02F5                 JP      Z,OUTP          ;OUTPUT PORT
00B2    18 95                   JR      MON
                        ;
00B4    DB 00           BREAK:  IN      A,(PIO)         ;BREAK CHECK
00B6    E6 80                   AND     80H
00B8    07                      RLCA
00B9    3F                      CCF
00BA    C9                      RET
                        ;
00BB    3E 0D           CRLF:   LD      A,0DH
00BD    CD 00E3                 CALL    SOUT
00C0    3E 0A                   LD      A,0AH
00C2    18 1F                   JR      SOUT
                        ;
00C4    3E 20           SPC:    LD      A,' '
00C6    18 1B                   JR      SOUT
                        ;
00C8    7E              DOUT:   LD      A,(HL)          ;(HL) HEX OUT
00C9    18 05                   JR      HXOUT
                        ;
00CB    7C              AOUT:   LD      A,H             ; HL  HEX OUT
00CC    CD 00D0                 CALL    HXOUT
00CF    7D                      LD      A,L
                        ;
00D0    F5              HXOUT:  PUSH    AF              ;HEX BYTE OUT
00D1    0F                      RRCA
00D2    0F                      RRCA
00D3    0F                      RRCA
00D4    0F                      RRCA
00D5    CD 00D9                 CALL    HOUT1
00D8    F1                      POP     AF
00D9    E6 0F           HOUT1:  AND     0FH
00DB    FE 0A                   CP      0AH
00DD    38 02                   JR      C,HOUT2
00DF    C6 07                   ADD     A,7
00E1    C6 30           HOUT2:  ADD     A,30H
                        ;
```

```
00E3    F3              SOUT:   DI
00E4    F5                      PUSH    AF              ;OUTPUT DATA SAVE
00E5    08                      EX      AF,AF'            ;OUTPUT DATA SAVE
00E6    DB 00           SOUT1:  IN      A,(PIO)
00E8    CB 4F                   BIT     1,A             ;CTS ON ?
00EA    20 FA                   JR      NZ,SOUT1
00EC    E6 FE                   AND     0FEH
00EE    D3 00                   OUT     (PIO),A         ;START BIT OUTPUT
00F0    D9                      EXX
00F1    21 011F                 LD      HL,CNST1
00F4    CD 0183                 CALL    BSET
00F7    CD 0192                 CALL    DELAY1          ;START BIT DELAY
00FA    CD 018B                 CALL    BSET1           ;DELAY CONST.->DE
00FD    DB 00                   IN      A,(PIO)
00FF    E6 72                   AND     72H
0101    6F                      LD      L,A             ;USER DATA->L
0102    08                      EX      AF,AF'
0103    4F                      LD      C,A             ;OUTPUT DATA->C
0104    06 09                   LD      B,9
0106    00                      NOP
0107    37                      SCF
0108    7D              SOUT2:  LD      A,L
0109    CB 19                   RR      C               ;1 BIT->CY
010B    CE 00                   ADC     A,0
010D    D3 00                   OUT     (PIO),A         ;1 BIT OUTPUT
010F    CD 0192                 CALL    DELAY1          ;FULL BIT DELAY
0112    10 F4                   DJNZ    SOUT2
0114    06 07                   LD      B,7
0116    10 FE           SOUT3:  DJNZ    SOUT3
0118    CD 0195                 CALL    DELAY2
011B    F1                      POP     AF              ;RECOVER ACC
011C    D9                      EXX                     ;RECOVER REG.
011D    FB                      EI
011E    C9                      RET
                        ;
011F    0397            CNST1:  DEFW    919             ;110 BPS
0121    039F                    DEFW    927
                        ;
0123    0149                    DEFW    329             ;300 BPS
0125    0151                    DEFW    337
                        ;
0127    0049                    DEFW    73              ;1200 BPS
0129    0051                    DEFW    81
                        ;
012B    0009                    DEFW    9               ;4800 BPS
012D    0011                    DEFW    17
                        ;
012F    C5              SEIN:   PUSH    BC
0130    DB 00                   IN      A,(PIO)
0132    E6 31                   AND     31H
0134    D3 00                   OUT     (PIO),A         ;RTS ON
0136    E6 30                   AND     30H
0138    08                      EX      AF,AF'            ;USER DATA SAVE
0139    D9                      EXX
013A    DB 00           SEIN1:  IN      A,(PIO)
013C    E6 80                   AND     80H
013E    C2 013A                 JP      NZ,SEIN1        ;START BIT ?
0141    21 019E                 LD      HL,CONST
0144    CD 0183                 CALL    BSET
0147    CD 0195                 CALL    DELAY2          ;HALF BIT DELAY
014A    CD 018B                 CALL    BSET1           ;DELAY CONST.->DE
014D    DB 00                   IN      A,(PIO)
014F    CB 7F                   BIT     7,A
0151    20 E7                   JR      NZ,SEIN1        ;START BIT ?
0153    08                      EX      AF,AF'
0154    6F                      LD      L,A             ;USER DATA->L
0155    D3 00                   OUT     (PIO),A         ;START BIT ECHO BACK
0157    03                      INC     BC              ;NOP     6T
0158    06 08                   LD      B,8
015A    CD 0190                 CALL    DELAY           ;FULL BIT DELAY
015D    DB 00           SEIN2:  IN      A,(PIO)         ;DATA INPUT
015F    E6 80                   AND     80H
0161    07                      RLCA
0162    CB 19                   RR      C
0164    B5                      OR      L               ;USER DATA
0165    D3 00                   OUT     (PIO),A         ;ECHO BACK
0167    CD 0190                 CALL    DELAY           ;FULL BIT DELAY
```

```
016A    10 F1                   DJNZ    SEIN2
016C    06 02                   LD      B,2
016E    10 FE           SEIN3:  DJNZ    SEIN3           ;TIME DELAY
0170    3E 41           SEIN4:  LD      A,41H
0172    B5                      OR      L
0173    D3 00                   OUT     (PIO),A         ;STOP BIT OUTPUT
0175    06 07                   LD      B,7
0177    10 FE           SEIN5:  DJNZ    SEIN5           ;TIME DELAY
0179    CD 0190                 CALL    DELAY
017C    CD 0190                 CALL    DELAY
017F    79                      LD      A,C             ;DATA->A
0180    D9                      EXX
0181    C1                      POP     BC
0182    C9                      RET
                        ;
0183    DB 00           BSET:   IN      A,(PIO)         ;BAUD RATE->DE
0185    E6 0C                   AND     0CH
0187    4F                      LD      C,A
0188    06 00                   LD      B,0
018A    09                      ADD     HL,BC
018B    5E              BSET1:  LD      E,(HL)
018C    23                      INC     HL
018D    56                      LD      D,(HL)
018E    23                      INC     HL
018F    C9                      RET
                        ;
0190    A7              DELAY:  AND     A               ;NOP     4T
0191    D8                      RET     C               ;NOP     5T
0192    3A 0000         DELAY1: LD      A,(0)           ;NOP     13T
0195    D5              DELAY2: PUSH    DE
0196    1B              DELAY3: DEC     DE
0197    7A                      LD      A,D
0198    B3                      OR      E
0199    C2 0196                 JP      NZ,DELAY3       ;DE=0 ?
019C    D1                      POP     DE
019D    C9                      RET
                        ;
019E    01C8            CONST:  DEFW    456             ;110 BPS
01A0    039E                    DEFW    926
                        ;
01A2    00A1                    DEFW    161             ;300 BPS
01A4    0150                    DEFW    336
                        ;
01A6    0021                    DEFW    33              ;1200 BPS
01A8    0050                    DEFW    80
                        ;
01AA    0001                    DEFW    1               ;4800 BPS
01AC    0010                    DEFW    16
                        ;
01AE    D6 30           CKHEX:  SUB     30H
01B0    D8                      RET     C
01B1    FE 0A                   CP      0AH
01B3    38 07                   JR      C,CKHX1         ;0-9
01B5    D6 11                   SUB     11H
01B7    D8                      RET     C               ;:-?
01B8    C6 0A                   ADD     A,0AH
01BA    FE 10                   CP      10H
01BC    3F              CKHX1:  CCF
01BD    C9                      RET
                        ;
01BE    21 0000         GETAD:  LD      HL,0            ;GET ADDRESS -> HL
01C1    CD 012F         GTAD1:  CALL    SEIN
01C4    FE 0D                   CP      0DH
01C6    C8                      RET     Z               ;CR .. Z=1
01C7    FE 2C           GTAD2:  CP      ','
01C9    20 03                   JR      NZ,GTAD3
01CB    D6 00                   SUB     0
01CD    C9                      RET                     ;',' .. Z=0
01CE    CD 01AE         GTAD3:  CALL    CKHEX
01D1    30 04                   JR      NC,GTAD4
01D3    F1                      POP     AF              ;DUMMY POP
01D4    C3 0086                 JP      ERR
01D7    29              GTAD4:  ADD     HL,HL
01D8    29                      ADD     HL,HL
01D9    29                      ADD     HL,HL
01DA    29                      ADD     HL,HL
01DB    85                      ADD     A,L
```

```
01DC    6F                      LD      L,A
01DD    18 E2                   JR      GTAD1
                        ;
01DF    CD 01BE         MEM:    CALL    GETAD           ; 'M' COMMAND
01E2    CA 0086                 JP      Z,ERR
01E5    EB                      EX      DE,HL           ;FROM START ADDRESS
01E6    CD 01BE                 CALL    GETAD           ;FROM END   ADDRESS
01E9    CA 0086                 JP      Z,ERR
01EC    A7                      AND     A
01ED    ED 52                   SBC     HL,DE
01EF    DA 0086                 JP      C,ERR           ;DE>HL .. ERROR
01F2    23                      INC     HL
01F3    44                      LD      B,H             ;BYTES = BC
01F4    4D                      LD      C,L
01F5    CD 01BE                 CALL    GETAD           ;TO ADDRESS
01F8    C2 0086                 JP      NZ,ERR
01FB    E5                      PUSH    HL
01FC    EB                      EX      DE,HL           ;FROM(HL) TO(DE) BYTES(BC)
01FD    F7                      RST     30H             ;CALL COMP
01FE    38 06                   JR      C,MEM1
0200    ED B0                   LDIR
0202    E1                      POP     HL
0203    C3 0049                 JP      MON
                        ;
0206    09              MEM1:   ADD     HL,BC           ;DE>HL
0207    2B                      DEC     HL
0208    EB                      EX      DE,HL
0209    09                      ADD     HL,BC
020A    2B                      DEC     HL
020B    EB                      EX      DE,HL
020C    ED B8                   LDDR
020E    E1                      POP     HL
020F    C3 0049                 JP      MON
                        ;
0212    CD 012F         DMP:    CALL    SEIN            ;DUMP MEMORY
0215    FE 0D                   CP      0DH
0217    28 19                   JR      Z,DMP2
0219    FE 2C                   CP      ','
021B    28 08                   JR      Z,DMP1
021D    21 0000                 LD      HL,0
0220    CD 01CE                 CALL    GTAD3           ;START ADDRESS
0223    28 0D                   JR      Z,DMP2
0225    EB              DMP1:   EX      DE,HL
0226    CD 01BE                 CALL    GETAD           ;END ADDRESS
0229    C2 0086                 JP      NZ,ERR
022C    F7                      RST     30H
022D    DA 0086                 JP      C,ERR
0230    18 05                   JR      DMP3
0232    EB              DMP2:   EX      DE,HL
0233    21 003F                 LD      HL,3FH
0236    19                      ADD     HL,DE
0237    EB              DMP3:   EX      DE,HL
0238    CD 00BB         DMP4:   CALL    CRLF
023B    CD 00CB                 CALL    AOUT            ;ADDRESS OUT
023E    CD 00C4                 CALL    SPC
0241    CD 00C4         DMP5:   CALL    SPC
0244    CD 00C8                 CALL    DOUT            ;DATA OUT
0247    CD 00B4                 CALL    BREAK           ;BREAK CHECK
024A    DA 0049                 JP      C,MON
024D    23                      INC     HL
024E    EB                      EX      DE,HL
024F    F7                      RST     30H
0250    EB                      EX      DE,HL
0251    DA 0049                 JP      C,MON           ; HL > DE
0254    7D                      LD      A,L
0255    E6 0F                   AND     0FH
0257    20 E8                   JR      NZ,DMP5
0259    18 DD                   JR      DMP4
                        ;
025B    CD 012F         SETM:   CALL    SEIN            ;SET MEMORY
025E    FE 0D                   CP      0DH
0260    28 09                   JR      Z,STM1
0262    21 0000                 LD      HL,0H
0265    CD 01CE                 CALL    GTAD3
0268    C2 0086                 JP      NZ,ERR
026B    CD 00BB         STM1:   CALL    CRLF
```

```
026E    CD 00CB                 CALL    AOUT
0271    CD 00C4                 CALL    SPC
0274    CD 00C4        STM2:    CALL    SPC
0277    CD 00C8                 CALL    DOUT
027A    3E 2D                   LD      A,'-'
027C    CD 00E3                 CALL    SOUT
027F    CD 012F                 CALL    SEIN
0282    FE 0D                   CP      0DH
0284    CA 0049                 JP      Z,MON           ;END
0287    FE 08                   CP      08H
0289    28 2B                   JR      Z,STM5          ;BACK SPACE .. ADDRESS DEC.
028B    FE 20                   CP      ' '
028D    20 05                   JR      NZ,STM3
028F    CD 00C4                 CALL    SPC             ;SPACE .. NO CHANGE
0292    18 1A                   JR      STM4
0294    CD 01AE        STM3:    CALL    CKHEX
0297    DA 0086                 JP      C,ERR
029A    07                      RLCA
029B    07                      RLCA
029C    07                      RLCA
029D    07                      RLCA
029E    4F                      LD      C,A
029F    CD 012F                 CALL    SEIN
02A2    CD 01AE                 CALL    CKHEX
02A5    DA 0086                 JP      C,ERR
02A8    81                      ADD     A,C
02A9    77                      LD      (HL),A
02AA    BE                      CP      (HL)            ;WRITE CHECK
02AB    C2 0086                 JP      NZ,ERR
02AE    23             STM4:    INC     HL              ;ADDRESS INC.
02AF    7D                      LD      A,L
02B0    E6 07                   AND     07H
02B2    28 B7                   JR      Z,STM1
02B4    18 BE                   JR      STM2
02B6    3E 2D          STM5:    LD      A,'-'           ;ADDRESS DEC.
02B8    CD 00E3                 CALL    SOUT
02BB    2B                      DEC     HL
02BC    18 AD                   JR      STM1
                       ;
02BE    CD 01BE        FILM:    CALL    GETAD           ;FILL MEMORY WITH CONSTANT
02C1    CA 0086                 JP      Z,ERR
02C4    EB                      EX      DE,HL           ;DE = START ADDRESS
02C5    CD 01BE                 CALL    GETAD
02C8    CA 0086                 JP      Z,ERR
02CB    F7                      RST     30H
02CC    DA 0086                 JP      C,ERR           ;DE>HL .. ERROR
02CF    E5                      PUSH    HL
02D0    CD 01BE                 CALL    GETAD
02D3    C2 0086                 JP      NZ,ERR
02D6    7D                      LD      A,L
02D7    E1                      POP     HL              ;HL = END ADDRESS
02D8    77             FLM1:    LD      (HL),A
02D9    F7                      RST     30H
02DA    CA 0049                 JP      Z,MON
02DD    2B                      DEC     HL
02DE    18 F8                   JR      FLM1
                       ;
                       ;                                ;INPUT PORT
02E0    EB             INP:     EX      DE,HL           ;CURRENT ADDRESS SAVE
02E1    CD 01BE                 CALL    GETAD
02E4    C2 02EA                 JP      NZ,INP1
02E7    CD 00BB                 CALL    CRLF
02EA    44             INP1:    LD      B,H
02EB    4D                      LD      C,L
02EC    ED 78                   IN      A,(C)
02EE    CD 00D0                 CALL    HXOUT
02F1    EB                      EX      DE,HL           ;CURRENT ADDRESS RECOVER
02F2    C3 0049                 JP      MON
                       ;
                       ;                                ;OUTPUT PORT
02F5    EB             OUTP:    EX      DE,HL           ;ADDRESS SAVE
02F6    CD 01BE                 CALL    GETAD
02F9    CA 0086                 JP      Z,ERR
02FC    44                      LD      B,H
02FD    4D                      LD      C,L
02FE    CD 01BE                 CALL    GETAD
```

```
0301    C2 0086                 JP      NZ,ERR
0304    ED 69                   OUT     (C),L
0306    EB                      EX      DE,HL           ;ADDRESS RECOVER
0307    C3 0049                 JP      MON
                        ;
030A    E3              LINOUT: EX      (SP),HL         ;PRINT STRINGS
030B    7E                      LD      A,(HL)
030C    23                      INC     HL
030D    E3                      EX      (SP),HL
030E    FE FF                   CP      0FFH            ;FF .. END
0310    C8                      RET     Z
0311    CD 00E3                 CALL    SOUT
0314    18 F4                   JR      LINOUT
                        ;
0316    EB              XMINR:  EX      DE,HL           ;EXAMINE REG.
0317    21 000C                 LD      HL,0CH
031A    39                      ADD     HL,SP
031B    CD 030A                 CALL    LINOUT
031E    0D 0A                   DEFB    0DH,0AH
0320    20 50 43 20             DEFB    ' PC '
0324    20 20 41 20             DEFB    '  A '
0328    46 20 20 42             DEFB    'F  B'
032C    20 43 20 20             DEFB    ' C  '
0330    44 20 45 20             DEFB    'D E '
0334    20 48 20 4C             DEFB    ' H L'
0338    20 20 53 50             DEFB    '  SP'
033C    0D 0A                   DEFB    0DH,0AH
033E    FF                      DEFB    0FFH
033F    06 06                   LD      B,6
0341    2B              XMNR1:  DEC     HL
0342    CD 00C4                 CALL    SPC
0345    CD 00C8                 CALL    DOUT
0348    2B                      DEC     HL
0349    CD 00C8                 CALL    DOUT
034C    10 F3                   DJNZ    XMNR1
034E    EB                      EX      DE,HL
034F    C3 0049                 JP      MON
                        ;
0352    21 0FFF         MSRCH:  LD      HL,0FFFH        ;RAM SERCH ROUTINE
0355    06 00                   LD      B,0             ;B-REG .. FIND FLAG
0357    23              SRCH1:  INC     HL
0358    7E                      LD      A,(HL)
0359    2F                      CPL
035A    77                      LD      (HL),A
035B    BE                      CP      (HL)
035C    20 0F                   JR      NZ,SRCH2
035E    2F                      CPL
035F    77                      LD      (HL),A
0360    BE                      CP      (HL)
0361    20 0A                   JR      NZ,SRCH2
0363    78                      LD      A,B
0364    A7                      AND     A
0365    20 F0                   JR      NZ,SRCH1
0367    54                      LD      D,H             ;RAM START ADDRESS -> DE
0368    5D                      LD      E,L
0369    06 FF                   LD      B,0FFH          ;FIND FLAG ON
036B    18 EA                   JR      SRCH1
                        ;
036D    78              SRCH2:  LD      A,B
036E    A7                      AND     A
036F    20 06                   JR      NZ,SRCH3
0371    7C                      LD      A,H
0372    A5                      AND     L               ;HL=FFFFH ?
0373    3C                      INC     A
0374    20 E1                   JR      NZ,SRCH1
0376    76                      HALT                    ;NO RAMS
                        ;
0377    F9              SRCH3:  LD      SP,HL           ;STACK POINTER SET
0378    CD 030A                 CALL    LINOUT
037B    0D 0A                   DEFB    0DH,0AH
037D    52 41 4D 20             DEFB    'RAM '
0381    41 52 45 41             DEFB    'AREA'
0385    20 20 20                DEFB    '   '
0388    FF                      DEFB    0FFH
0389    EB                      EX      DE,HL
038A    E5                      PUSH    HL
```

```
038B    CD 00CB              CALL   AOUT          ;RAM START ADDRESS
038E    CD 030A              CALL   LINOUT
0391    20 2E 2E 20          DEFB   ' .. '
0395    FF                   DEFB   0FFH
0396    EB                   EX     DE,HL
0397    2B                   DEC    HL            ;RAM END ADDRESS
0398    CD 00CB              CALL   AOUT
039B    CD 00BB              CALL   CRLF
039E    E1                   POP    HL
                     ;
039F    11 0800      CKSAM:  LD     DE,800H
03A2    3A 0FFF              LD     A,(0FFFH)
03A5    FE 55                CP     55H
03A7    20 03                JR     NZ,CKSM1
03A9    11 1000              LD     DE,1000H
03AC    21 0000      CKSM1:  LD     HL,0
03AF    AF                   XOR    A
03B0    86           CKSM2:  ADD    A,(HL)
03B1    23                   INC    HL
03B2    F7                   RST    30H
03B3    20 FB                JR     NZ,CKSM2
03B5    A7                   AND    A
03B6    28 19                JR     Z,CKSM3
03B8    CD 030A              CALL   LINOUT
03BB    52 4F 4D 20          DEFB   'ROM '
03BF    43 48 45 43          DEFB   'CHEC'
03C3    4B 53 41 4D          DEFB   'KSAM'
03C7    20 45 52 52          DEFB   ' ERR'
03CB    4F 52 21             DEFB   'OR!'
03CE    0D 0A FF             DEFB   0DH,0AH,0FFH
                     ;
03D1    7A           CKSM3:  LD     A,D
03D2    FE 10                CP     10H
03D4    CA 0800              JP     Z,800H
03D7    FF                   RST    38H

                             END


Macros:

Symbols:
00CB    AOUT         00B4    BREAK        0183    BSET
018B    BSET1        01AE    CKHEX        01BC    CKHX1
039F    CKSAM        03AC    CKSM1        03B0    CKSM2
03D1    CKSM3        011F    CNST1        0030    COMP
019E    CONST        00BB    CRLF         0190    DELAY
0192    DELAY1       0195    DELAY2       0196    DELAY3
0212    DMP          0225    DMP1         0232    DMP2
0237    DMP3         0238    DMP4         0241    DMP5
00C8    DOUT         0000    ENTRY        0086    ERR
02BE    FILM         0208    FLM1         01BE    GETAD
01C1    GTAD1        01C7    GTAD2        01CE    GTAD3
01D7    GTAD4        0057    GUSR         00D9    HOUT1
00E1    HOUT2        00D0    HXOUT        02E0    INP
02EA    INP1         030A    LINOUT       01DF    MEM
0206    MEM1         0049    MON          008F    MON1
0352    MSRCH        0066    NMI          006E    NMI1
02F5    OUTP         0000    PIO          0038    RST38
0072    RTN          012F    SEIN         013A    SEIN1
015D    SEIN2        016E    SEIN3        0170    SEIN4
0177    SEIN5        025B    SETM         00E3    SOUT
00E6    SOUT1        0108    SOUT2        0116    SOUT3
00C4    SPC          0357    SRCH1        036D    SRCH2
0377    SRCH3        026B    STM1         0274    STM2
0294    STM3         02AE    STM4         02B6    STM5
0316    XMINR        0341    XMNR1
```

参考文献

(1) 猪飼國夫：インターフェース回路の設計，1980，CQ出版社

(2) 野崎　真：インターフェース設計入門，1983，ナツメ社

(3) Zilog：*Microcomputer Components Data Book,* 1981

(4) Intel：*Component Data Catalog,* 1982

(5) T.I.：*The Bipolar Integrated Circuits Data Book,* 1981，*part 1, part 2*

(6) Analog Devices：*Data-Acquisition Databook,* 1982

(7) Intersil：*Data Book,* 1982

(8) N.S.：*Linear Databook,* 1982

(9) RCA：*COS/MOS Integrated Circuits,* 1980

(10) N.S.：*CMOS Databook,* 1981

(11) 日立：HD46505S ユーザーズ マニュアル

(12) 日立：HD44780 ユーザーズ マニュアル

(13) Motorola：*CMOS Data,* 1981

(14) Siliconix：*Analog Switch & IC Product Data Book,* 1982

(15) 松下電工：ナショナル電子回路用リレー，1983

(16) Intel：*iAPX 88 Book,* 1981

索　引

著者略歴

畔津 明仁（あぜつ あきひろ）

1954年　福岡県で生まれる

1977年　東京工業大学工学部卒業

1979年　東京工業大学大学院にて修士号を取得

1982年　同大学院博士課程を中退

現　在　電子機器の設計・開発に従事

実用インターフェース設計法

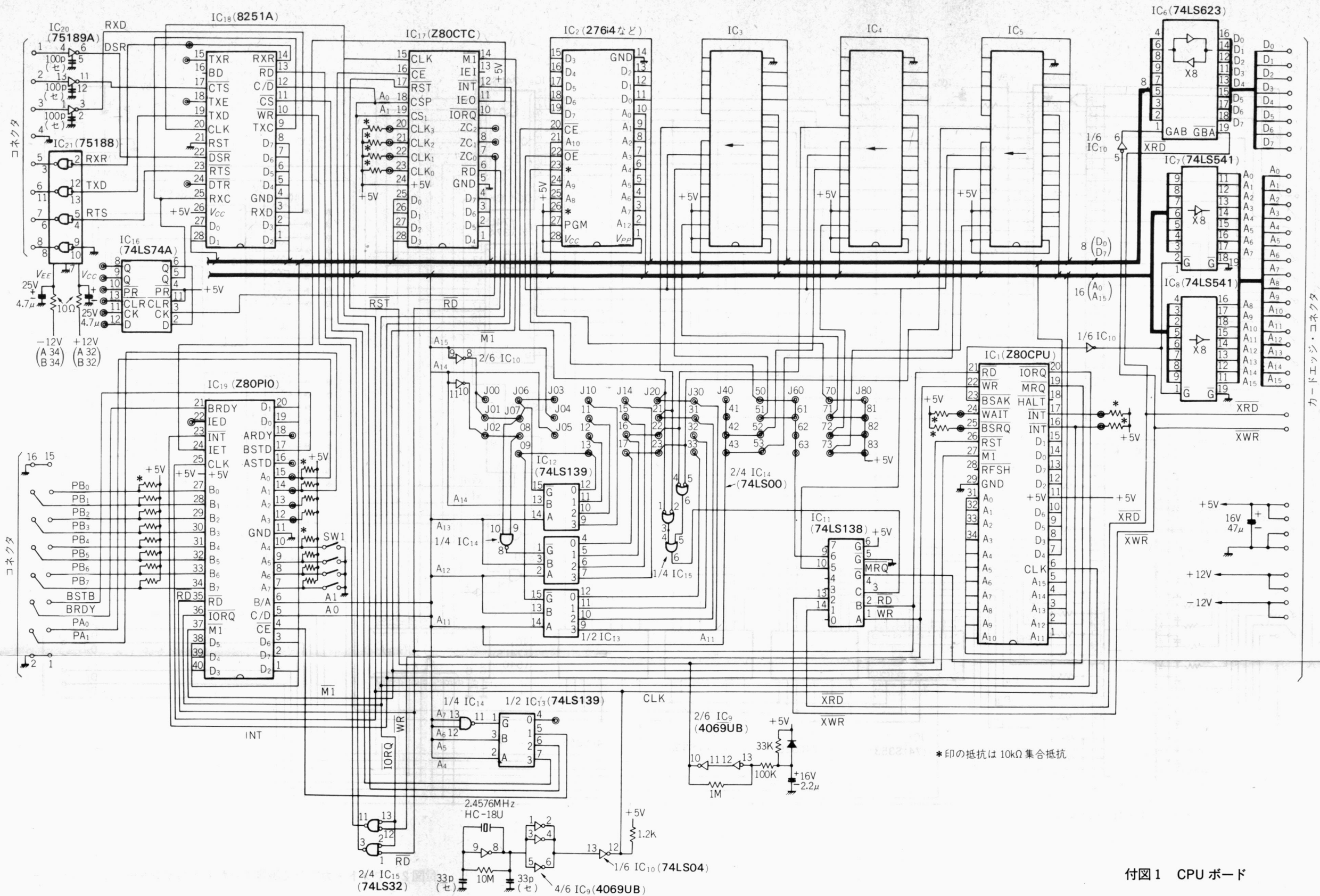
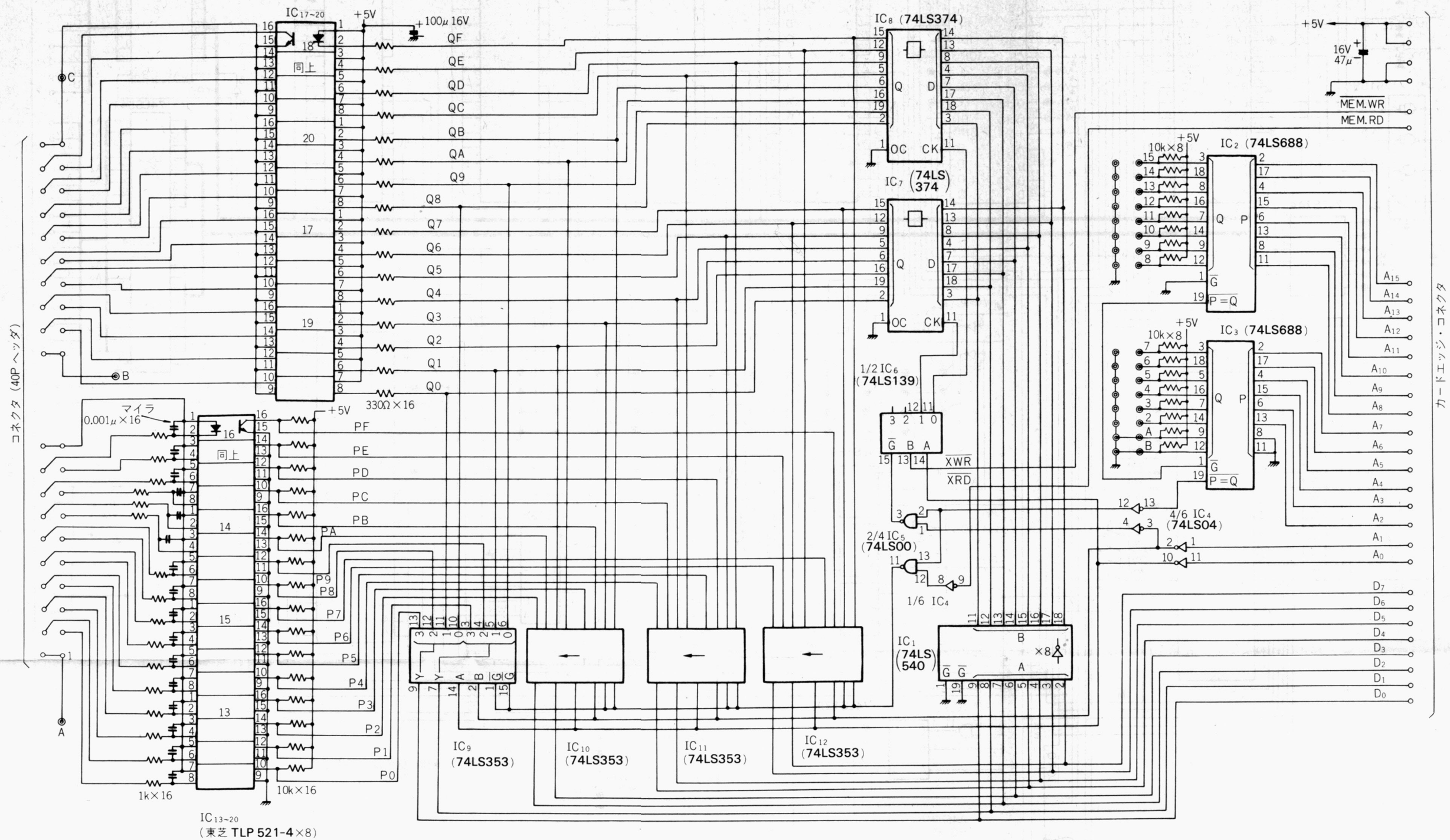
1985年3月30日　初版発行
1989年8月1日　第7版発行

著　者　畔　津　明　仁
発行人　神　戸　一　夫
発行所　CQ出版株式会社
東京都豊島区巣鴨1－14－2（〒170）
電話 03（947）6311（代）振替東京0-10665

（定価はカバーに表示してあります）

乱丁，落丁本はお取り替えします

制作・写植　都写植／印刷・製本　啓文堂

*印の抵抗は 10kΩ 集合抵抗

付図1　CPU ボード

付図２　フォト・カプラで絶縁された入出力インターフェース・ボード

付図3　操作箱用インターフェース・ボード

付図４　モータ制御用ボード